

# PRAGMA

A publication of Semaphore Corporation

Issue Number 6

November, 1983

## IN THIS ISSUE

|                                                            |    |
|------------------------------------------------------------|----|
| The Ubiquitous POINTER-FILE .....                          | 6  |
| An Introduction to ENGLISH®, Part 5: Syntax Overview ..... | 12 |
| Designing Data Entry Programs .....                        | 22 |
| GET, Part 1: An Input Processor .....                      | 28 |
| Vanilla, Part 6: Inspection .....                          | 34 |
| Individual Accounts or Shared Accounts? .....              | 40 |
| Lock Logic Illustrated .....                               | 42 |
| Two Undocumented Conversions .....                         | 44 |

---

## DEPARTMENTS

|                         |    |                         |    |
|-------------------------|----|-------------------------|----|
| Utilities .....         | 8  | Command Files .....     | 32 |
| Producer Profile .....  | 14 | Queries .....           | 39 |
| Benchmarks .....        | 20 | The Computer Room ..... | 43 |
| Wish List .....         | 27 | Letters .....           | 45 |
| Local User Groups ..... | 31 | Games .....             | 46 |



Your PC can now be a "Personal Minicomputer" with the **Revelation Operating System** from Cosmos.

**Revelation** is a Relational Data Base Management System for the Personal Computer that provides minicomputer power at microcomputer prices. You can now have a small, portable computer that will serve as a working tool on the job, at home or on the road for less than \$5,000.

**Revelation** is PICK™ compatible and operates on the IBM™ Personal Computer, as well as the Columbia Multi-Personal™ Computer and the COMPAQ™.

As a Relational Data Base Management system, **Revelation** operates on top of MS-DOS™, allowing you to switch back and forth between either operating environment and utilize hundreds of existing programs.

Relational Data Base Management and **Revelation** provide easy access to your data on floppy or hard disk. You use meaningful field names to refer to the data you desire. You can also define relationships that exist between two or more fields.

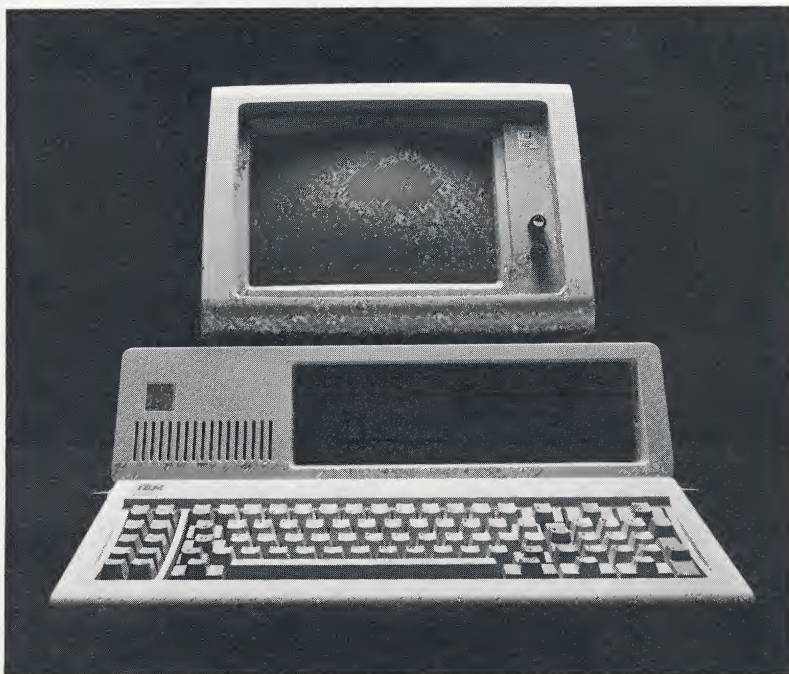
**Revelation** employs the Intel 8087 math chip on the PC for powerful number crunching capacity, floating point arithmetic and hardware mathematical functions.

Our application documentation and generation tools make **Revelation** ideal for software development and maintenance. And **Revelation's** advanced technology includes source code generation so you can do any job that needs to be done.

Communications problems between minicomputers and microcomputers are solved as **Revelation** allows your machine to communicate with compatible minicomputers.

Now you can afford the advantages of distributed processing.

# Why wait?



IBM Personal Computer

Collect data at a remote site during the day and ship it to your minicomputer when you want. You can also use your PC as a dumb terminal to initiate jobs on your local or remote minicomputer.

System Memory hassles are eliminated. Just add one of the many available 512K RAM boards to your microcomputer and you have the memory you need to do your work quickly and efficiently with **Revelation**.

Now you can have minicomputer capabilities at microcomputer prices. From Cosmos.

## Sample configuration

### Sample IBM System

- System unit, 64K RAM
- keyboard, 160K disk drive, and disk drive adapter

#### System price

- |                                          |                |
|------------------------------------------|----------------|
| • 2nd 160KB disk drive                   | \$2,205        |
| • Monochrome display                     | \$450          |
| • Monochrome display and printer adapter | \$345          |
| • 8087 math chip                         | \$335          |
| • 256K RAM                               | \$220          |
| with serial interface                    | \$349          |
| • Revelation software                    | \$950          |
| • MS-DOS                                 | \$40           |
| <b>Total</b>                             | <b>\$4,894</b> |

### Minimum configuration

- 320K RAM
- 8087 math chip
- MS-DOS or PC-DOS
- One single-sided floppy drive

# COSMOS

P.O. Box AH  
123 Ferntree Drive W.  
Morton, WA 98356  
(206) 496-5974  
24-hour answering service: (206) 226-9362

MS/DOS™ of Microsoft Corp., Intel™ of Intel Corp., ADDS Mentor™ of Applied Digital Data Systems, PICK Operating System™ of PICK SYSTEMS, COMPAQ Portable Computer™ of COMPAQ Computer Corp., Columbia Multi-Personal Computer is registered™ of Columbia Data Products Inc.



# Pick Pie Pictured

*Just how widespread is it?* Vendors and users researching the Pick operating system who contact us probably ask that question more than any other. *How many different machines are there for each implementation?* That's another popular question. Everyone has estimates and guesses, but no one really seems to be a truly knowledgeable authority who can quote exact figures. Bell Labs and Western Electric seem to know exactly how many Unix sites are licensed. The University of California at San Diego and its UCSD Pascal licensees seem to be keeping good track of their customers. And so on with other portable operating systems. Unfortunately, the rights to selling Pick and Pick-style systems have for some time been in the hands of a wide variety of vendors who are under no requirement to reveal their customer base. Numbers quoted in the past always seem to be rounded at least to the nearest hundred, and often sound more like they are rounded to the

nearest thousand. Interestingly, if every vendor takes more or less the same approach in rounding or even exaggerating their figures, each vendor's share of the market might still be represented with some accuracy, even if the exact total market size is not perfectly known.

With that rationalization in mind, we present the chart below as our estimate of how the U.S. Pick-style market is currently shared. It is based on claims, quotes, and our best guess.

...

For some months now, the staff has been calling this month's Pragma "the graphics issue". The pie chart is only the first of a number of diagrams and illustrations the reader will notice in the pages that follow. In the past, Pragma has always heavily relied on words to describe concepts and results. The graphics in this issue are an experiment to see just how well we can make pictures do the same job. We invite your comments.

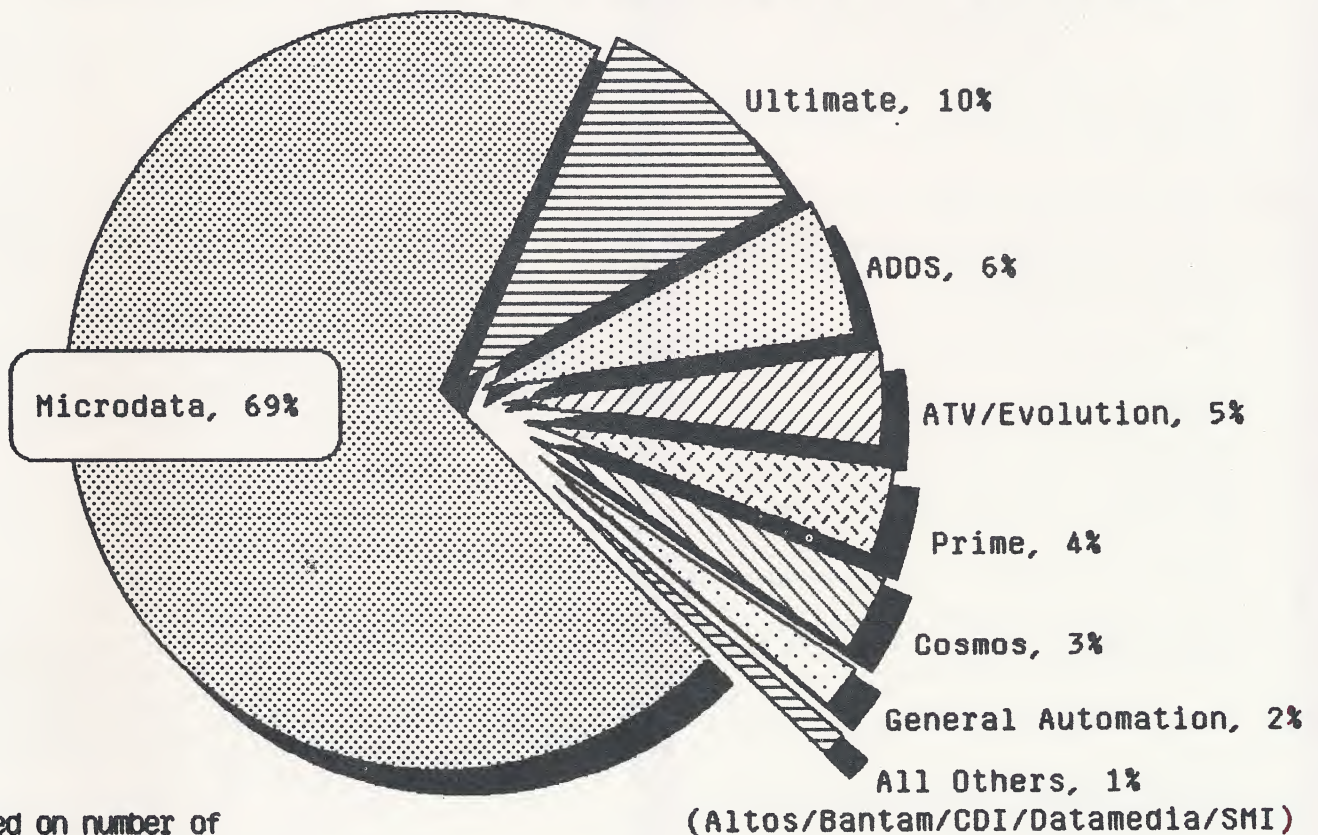
...

New entry #8000 on our mailing list and winner of the customary box of See's chocolates: Daniel Rayner of Lovelace Health Plan in Albuquerque.

— The Editors

P

## Estimated USA Pick-Style Market Percentages



Based on number of machines shipped.



# PRAGMA

Issue Number 6

November, 1983

Pragma is published quarterly by

Semaphore Corporation  
207 Granada Drive  
Aptos, California 95003

Entire contents copyright © 1983 by Semaphore Corporation. All rights reserved. No part of this journal may be reproduced, transmitted, transcribed, stored in a recording, retrieval or computer system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Semaphore Corporation.

Semaphore Corporation offers no warranty, either expressed or implied, for any losses due to the use of any material published in Pragma.

---

Subscriptions are \$25 per issue in the USA, \$38 per issue to other countries by airmail. All payments must be in US dollars drawn on a US bank.

Address all subscription correspondence to the Pragma Circulation Manager, Semaphore Corporation. When writing, enclose your issue's mailing address label or the numbers from the upper right corner of your label.

---

Address changes should be sent at least four weeks in advance. Include your new address along with your issue's mailing address label.

---

All correspondence and material received will be considered for publication. Address all submittals and correspondence to the Pragma Editors, Semaphore Corporation. All letters to the Editors are welcome and as many as possible will be published in the Letters Department. The Editors reserve the right to edit all submittals.

---

Is there a service Semaphore Corporation can provide for you? Address all inquiries to Semaphore Corporation or telephone 408-688-9200.

---

MENTOR is a trademark of Applied Digital Data Systems • ALL, DATA/BASIC, ENGLISH, MICRODATA, PRISM, REALITY, REFLEX, ROYALE, RUNOFF, SCREENPRO, SEQUEL, WORDMATE are trademarks of Microdata Corp. • INFORMATION is a trademark of Prime Inc. • ULTIMATE is a trademark of Ultimate Corp.

## Back Issues Are Available!

**Pragma #1, August 1982:** ZIP Code File Design • A Program for Renumbering Statement Labels • Deriving Year-to-Date Counts • VANILLA, The No-Frills Manufacturing System: The Parts File • Moletron User Profile • More Memory, Fewer Reads • SYSMAP: A Cross-Reference System • How to Flag Missing Checks • 23 Wish List Items • Computing Modulos with ENGLISH® • Building ENGLISH Headings with Proc • 4 Queries • Making Item Identifiers Hash Well • Tape Handling • Patching the Exit Problem in 3.2 SCREENPRO™ • The Shell Game.

**Pragma #2, November 1982:** A Modulo Setting Program • Black Box Formatting • TRIM.DELIM and PROFILE Utilities • SYSMAP, A Cross-Reference System: File Format Input • Galinski Hamburg User Profile • LOOP vs. LOCATE • 25 Wish List Items • An Introduction to ENGLISH: Jargon • Proc Conversions • VANILLA, The No-Frills Manufacturing System: Bill of Material Input • 14 Queries • The Trouble Tree • 2 Letters • A Switchbox for 32 Ports • Security and the DATA/BASIC™ Programmer • The Cookie Game • Some New Subscribers.

**Pragma #3, February 1983:** Edit Aides • A Proc for Cross-Referencing Q-Pointers • SYSMAP, A Cross-Reference System: Remaining File Input • Rainbow Natural Foods User Profile • More BASIC Benchmark Comparisons • Justifying Ragged Output • 13 Wish List Items • Generating Monthly Column Headings • VANILLA, The No-Frills Manufacturing System: Purchasing • 5 Queries • An Introduction to ENGLISH: More Commands • Shared Site Checklist • Converting Paint to Programs • 3 Letters • Generating Blank Forms • Animal, A Game that Learns • More New Subscribers.

**Pragma #4, May 1983:** Pacific Valley Bank User Profile • SYSMAP, A Cross-Reference System: Dicts and Procs • Uncompiling: Unassembling Stack Code • Left vs. Right Justification Benchmarks • 4 Wish List Items • A Comparison of BASIC Implementations • More New Subscribers • An Undocumented Editor Capability • 3 Local User Group Reports • Avoiding Saved Lists with PQ-RESELECT • Self-Documenting Reports • A Query • A Survey of Hardware that Supports Pick-Style Software • Printer Trade-Offs • An Introduction to ENGLISH: Finding files • A Letter • VANILLA, The No-Frills Manufacturing System: Purchase Order Entry • The Swat! Game.

**Pragma #5, August 1983:** Interactive Systems Producer Profile • Uncompiling: Regenerating Source Code • A Day of Revelation • IBM Personal Computer vs. Microdata™ Reality® Benchmarks • VANILLA, The No-Frills Manufacturing System: Receiving • 3 Wish List Items • An Introduction to ENGLISH: Being Choosy • Converting Manual Paint to Programs • 6 Local User Group Reports • More New Subscribers • A Program that Reports File Pointer Locations • Boiler Plate Processing with Runoff™ • A New Query and an Old Query Answered • SYSMAP, A Cross-Reference System: Automation • Tape Types • 6 Letters • Permuted Index to the First Four Issues of Pragma • Amazing, a Maze Game.

Send \$25 for each back issue to:

**Pragma**  
207 Granada Drive  
Aptos, CA 95003



# GAAP<sup>TM</sup> II

## Financial Management

# The Software Package You Can Account On.

GAAP Financial Software ... the only accounting tool you will need to make management decisions. Accurate. Flexible. User-friendly. KDK's software system is fully integrated with built-in security control. Automatic is the key word. The components of GAAP include:

### GENERAL LEDGER

The ultimate bookkeeper supplies comprehensive audit trails by accounting and entry date, summary and working trial balances, detailed general and subsidiary ledgers and unlimited USER-DEFINED financial statement formats. All reports may access ANY accounting periods at any time. Statements allow flexible reporting of net, budget, budget variance and comparative percentage ratios for period-to-date and year-to-date activity on a cash or accrual basis ... with optional consolidation by company.

### ACCOUNTS PAYABLE

Track your payables and accruals accurately by vendor, voucher and G/L account. Automatic payment scheduling, manual and system-generated disbursements, cash requirements reporting and manual or tape interface bank reconciliation simplify payable processing.

### ACCOUNTS RECEIVABLE

Generate balance forward or open invoice statements, finance charges, aged trial balances and analysis reports for selected groups or all clients. Payment on account, open invoice and delinquent account processing assist in receivables turn around.

### PAYROLL

Manual as well as automatic check distribution to employees at multiple locations ... with job costing interfaces. GAAP standard payroll reporting takes W2 and multi-state taxing into account.

Other completely interfaceable products offered by KDK include: Purchasing/Receiving, Inventory Control, Hotel Management and Apartment Billing.

GAAP ... financial management software designed to enhance your PICK operating system. Complete and return the coupon below or call KDK today about discounts offered on our software through January 1984!

 KDK Enterprises, Inc.  
Suite 302  
1491 Chain Bridge Road  
McLean, VA 22101  
(703) 893-7883

I'd like to know more about GAAP—the accurate and flexible financial management software for my PICK operating system; please send your FREE brochure.

Name: \_\_\_\_\_ Company: \_\_\_\_\_ Title: \_\_\_\_\_ Phone: (\_\_\_\_) \_\_\_\_\_  
City: \_\_\_\_\_ Address: \_\_\_\_\_ State: \_\_\_\_\_ Zip: \_\_\_\_\_  
Computer: \_\_\_\_\_

 **KDK Enterprises, Inc.**  
Suite 302  
1491 Chain Bridge Road  
McLean, VA 22101  
(703) 893-7883



# The Ubiquitous POINTER-FILE

A map of the POINTER-FILE and related files is presented, and an explanation of these structures' use and contents is given.

Most users, especially operators, programmers and maintenance personnel, are aware that a file called POINTER-FILE exists on their system, but few really know what it does or why it is needed. The map on the opposite page, in conjunction with the following discussion, should provide some enlightenment. (Note that this article is based on the Reality® implementation. Users of other systems may note some differences on their particular machine.)

The SYSTEM file (A) serves as the root for the large tree of files, items and pointers that are stored on disk and make up the file system. (Capital letters in parentheses refer to structures on the map.) Among other things, the SYSTEM file contains two items named SYSPROG (B) and POINTER-FILE (C). Like other items in the SYSTEM file that have a D in attribute one, the SYSPROG item (B) contains a pointer to a Master Dictionary (D), which is just a file that users can be associated with when they log on under the SYSPROG account name. Although not shown on the map, every account has an item in the SYSTEM file which points to that account's Master Dictionary.

The concept of a *pointer* must be clear in order to understand what the POINTER-FILE does. Imagine a mail carrier who retrieves and deposits mail each day from a row of mail boxes. Each mail box has its address painted on the side, and each box can hold data (pieces of mail). Every day, the mail carrier opens each box to retrieve and deposit data in the boxes. But one day, the carrier opens up box 207 and finds a note that says, "Dear Mail Person. We are on vacation. Please deposit our mail at our friend's house, in box 414." This note is a pointer! It warns the carrier that data should not be placed in this box, but a different box, and gives its address. In other words, a box that stores the address of some other box is storing a pointer. The pointer is the address of the other box.

Items and frames on the disk are just like mail boxes. They all have addresses and they can store data. Items have arbitrary names as addresses, while frames have numbers as addresses. The SYSPROG item (B) in the SYSTEM file contains a pointer to a Master Dictionary (D). What is actually stored in the item (in attribute two) is the number, or address, of the disk frame where the Master Dictionary starts. The map shown here could include a frame number inside the SYSPROG box and the same number on the outside of the Master Dictionary box (just like mail boxes), but the convention normally used when documenting computer systems is to simply draw an arrow starting from where the pointer is stored and ending at the object it points at. That way, numbers don't have to be visually matched up.

Note that the D in attribute one of some items is what indicates to the operating system that the item is storing a pointer, and not some other kind of data. In the mail box example above, the warning "Dear Mail Person" told the carrier that the slip of paper was a pointer, and not just another piece of mail. In the same way, the D in attribute one warns the operating system that an item contains a pointer, and that a frame address can be found in attribute two. An example of an item in the SYSTEM file that does not have a D in attribute one, and which therefore does not contain a pointer, would be the LOGON item (not shown).

Just as the SYSPROG item in SYSTEM points to a set of frames called the SYSPROG Master Dictionary, the POINTER-FILE item (C) in SYSTEM points to a set of frames called the POINTER-FILE (E). Note that the item inside the SYSPROG Master Dictionary named POINTER-FILE (F) also contains a pointer to POINTER-FILE (so that users who log on to SYSPROG can use the word "POINTER-FILE" in their commands). Master Dictionary item (F) uses a Q in attribute one because the address stored in attributes two and three happens to be a name, not a number.

What is stored inside the POINTER-FILE (E) frames? More pointers! Each item in POINTER-FILE either has a D in attribute one (G), or a C in attribute one (H). The D item (which should only be the item named DL/ID) is a standard pointer just like those in the SYSTEM file. Its purpose in POINTER-FILE is to tell the operating system that this file has its dictionary and data all at one level (like a Master Dictionary), which is usually of no real concern to most users.

The POINTER-FILE items with a C in attribute one (like H and K) are created any time a program is cataloged or a SELECT list is saved. Imagine a program named SAMPLE that is cataloged by a user logged on to the SYSPROG account. To perform the catalog, the operating system immediately does three things: (1) unused disk frames (I) are allocated for storing the program's object code, (2) an item named SYSPROG\*C\*SAMPLE (H) is placed in POINTER-FILE with a C in attribute one, the number of the first frame of the program's object code in attribute two, the total number of frames of object code in attribute three, and the time and date the program was cataloged in attribute five, and (3) an item named SAMPLE (J) is stored in the Master Dictionary.

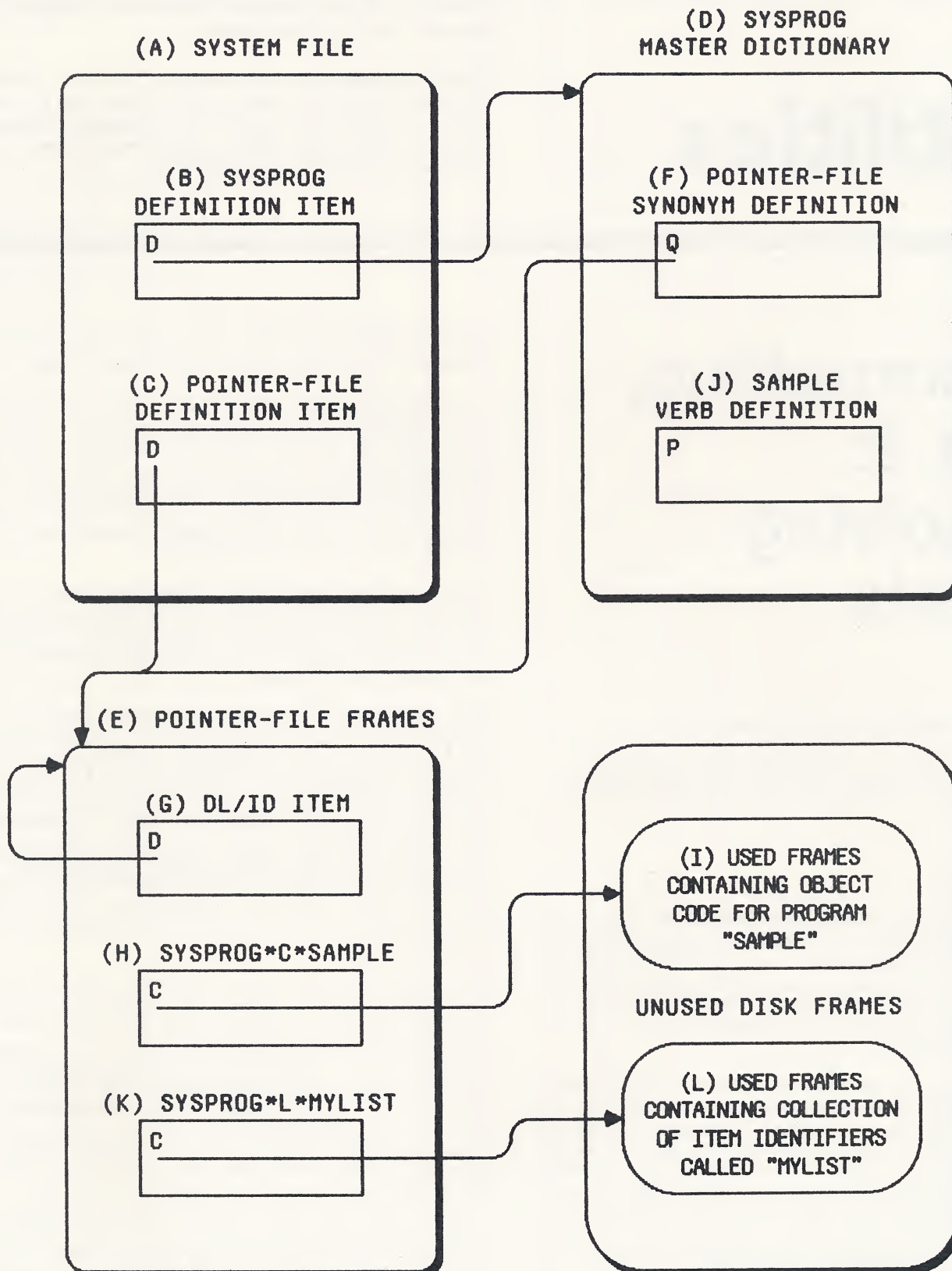
With those items in place, the operating system can now correctly execute the appropriate object code when the user gives the SAMPLE command: just as when any command is given, the Master Dictionary is searched to determine the verb type. Finding the SAMPLE item (J) eventually causes the POINTER-FILE item (H) to be located, which leads to the object code itself (I).

The POINTER-FILE items with \*L\* in their identifier (like K) are created any time a SELECT list is saved. Imagine a user logged on to the SYSPROG account who gives the command SAVE-LIST MYLIST. To perform the save, the operating system: (1) allocates unused disk frames (L) for storing the list of item identifiers, and (2) places an item named SYSPROG\*L\*MYLIST (K) in the POINTER-FILE with a C in attribute one, the number of the first frame containing the list in attribute two, the total number of frames occupied by the list in attribute three, the total number of items in the list in attribute four, and the time and date the list was saved in attribute five. Since list names are never used as verbs, no verb definition item like (J) needs to be stored in the Master Dictionary when a list is saved.

□



## The POINTER-FILE and Related Structures





# utilities

## Uncompiling, Part 3: Resolving Labels

The last in a series of articles devoted to uncompiling object code is presented. Portions of previous versions of un compilers are modified to allow statement labels and GO TOs to be properly resolved.

The un compilers presented so far [*Pragma #4, page 12* and *Pragma #5, page 6*] have not included capabilities for the proper handling of statement label numbers or for the handling of references to such labels, as in GO TO statements. How should the previously presented un compilers be modified so that statement labels and their references in object code are properly uncompiled?

While an un compiler scans through a program's object code, it can easily recognize and resolve a *forward reference*, such as a GO TO that references a statement further down in the program's code. When a forward reference is found, the un compiler simply has to remember the address of the referenced statement. When the referenced statement is finally reached, the un compiler knows the statement's address should be output as a label since some previous statement made a reference to the location. A *backward reference*, such as a GO TO that references a previous statement already scanned earlier, is a bit trickier to handle. One of three methods is generally used to allow backward references to be correctly uncompiled:

Utility programs have saved many a programmer from the less appealing aspects of software development and maintenance. Reformatting code, stripping files clean of control characters, converting data from one format to another — all are examples of tasks best delegated to a program and not a programmer.

Good programmers will collect a "toolbox" of utility software to use from time to time. If you have a useful utility, send it in for publication. A regular feature in *Pragma* will be this Utilities Department, where good software tools will be spotlighted.

1. **Output labels everywhere.** This is the easiest method for the un compiler, but generates the messiest output, since every statement, referenced or not, is prefixed with a label.

2. **Use a temporary file.** This scheme outputs the regenerated source statements to some type of file with, say, each line of source code set up as a separate file record, so that if a backward reference is encountered, the target statement's source code can be easily found in the file and prefixed with a label. This guarantees only one pass over the object code, but forces all of the output to be temporarily saved and not listed until all of the object code is scanned.

3. **Scan the object code twice.** This involves making a list of referenced statements during the first pass and printing all statements on the second pass, while including a label only if the statement being printed is in the list of statements found referenced by the first pass.

This series' third and final un compiler version is the LABEL.GEN program beginning on page 10. LABEL.GEN uses the last of the three methods described above, and is designed to only be concerned with the correct un compilation of GO TOs and their target statement label numbers. The code beginning at line 47 of LABEL.GEN, which is concerned with the proper handling of GO TO opcodes, can easily be duplicated to provide similar support for statement label references by opcodes such as GOSUB or RETURN TO, while the previously published un compilers have already demonstrated source code regeneration techniques for other statement types.

The sample program in original and uncompiled versions at the bottom of page 11 proves that LABEL.GEN correctly identifies statement boundaries while generating statement labels at appropriate positions so that all GO TOs are properly matched. (This version of the un compiler simply outputs "STATEMENT" if the object code is anything other than a GO TO.) For the few times that an un compiler should actually be needed, the reader is advised to simply process the mystery object code with each of the three un compiler versions that have been presented (the stack code un assembler from *Part 1*, the source code generating version from *Part 2*, and LABEL.GEN), and a fairly complete picture of the equivalent source program should then emerge. For high volume production un compiling, the techniques demonstrated by each of the three versions should be combined into one un compiler so that the most complete source code possible can be regenerated in one step.

P



# A picture is worth a thousand words.

ACCU-PLOT II from AccuSoft Enterprises is a new, easy to use graphics system that will produce bar charts, point-to-point line charts, scatter diagrams and pie charts in black and white or color.

ACCU-PLOT II is as easy to use as ENGLISH. The sentences used to produce graphs are similar to the "LIST" and "SORT" verbs you use every day. No modification to your dictionaries or data files is required.

ACCU-PLOT II is available for Microdata, Ultimate, ADDS Mentor, Evolution, CDI's IBM Series/1, Datamedia 932, General Automation Zebra and other PICK based computer systems. ACCU-PLOT II will operate on most dot matrix printers with graphics capabilities, including the IDS color Prism and other color graphics devices such as pen plotters.

**AccuSoft Enterprises**  
8043 Foothill Boulevard  
Sunland, California 91040

## You will benefit from ACCU-PLOT II because:

- ACCU-PLOT II provides a quick and easy way for you to evaluate your business, saving you time and money.
- ACCU-PLOT II creates business charts using ENGLISH so you will not have to learn complicated instructions.
- ACCU-PLOT II works with your existing data files, thus no modification to your system is needed.
- ACCU-PLOT II works with a standard Printronix printer, which means you don't have to purchase additional hardware.
- ACCU-PLOT II interfaces with many output devices for a variety of black and white and color applications.
- ACCU-PLOT II creates bar, line, scatter and pie charts, allowing for presentation of data in various formats.
- ACCU-PLOT II pie charts may have "exploded" slices used for highlighting your information.
- ACCU-PLOT II is available for \$995.00 with a 30 day free trial period, so you can use it for one month without any financial obligation.
- ACCU-PLOT II has step by step loading instructions, making it easy to install.
- ACCU-PLOT II interfaces with COMPU-SHEET electronic spreadsheet so you can represent your worksheets in graphics form.

**Put the graphic power of ACCU-PLOT II at your command. For more information, mail this coupon or call (213) 352-1233 today.**

**Yes! Please send me more information about ACCU-PLOT II.**

Name   
Title   
Company   
Address   
City   
State  Zip   
Telephone (  )



```

LABEL GEN
001 EQU EOL TO CHAR(1)
002 PRINT "FILE": ; INPUT FILE
003 OPEN FILE ELSE STOP "NO SUCH FILE!"
004 PRINT "ITEM": ; INPUT ITEM.ID
005 READ ITEM FROM ITEM.ID ELSE STOP "NO SUCH ITEM!"
006 DEL ITEM<1> ; FIRST.LINE = ITEM<1>
007 ITEM<1> = FIRST.LINE[6, LEN(FIRST.LINE)-5]
008 TOTAL.LINES = COUNT(ITEM, EOL)
009 IF TOTAL.LINES = 0 THEN TOTAL.LINES = 1
010 LABELS = ""
011 FOR PASS = 1 TO 2
012   LOC = 46
013   FOR I = 1 TO TOTAL.LINES
014     INSTRUCTS=FIELD(ITEM, EOL, I):EOL
015     PTR = 1
016     UNKNOWN = 0 ; ALREADY.PRINTED = 0 ; LABEL.NUM = LOC
017     LOOP
018       OPCODE=INSTRUCTS[PTR, 1]
019       UNTIL OPCODE="" DO
020         HEXCODE=OCONV(OPCODE, "MX")
021         PTR = PTR+1 ; LOC = LOC+1
022         GOSUB 100 ; * DECODE
023       REPEAT
024       IF PASS = 2 THEN
025         IF UNKNOWN THEN PRINT " UNKNOWN OPCODES!" ELSE
026         IF ALREADY.PRINTED THEN PRINT ELSE PRINT "*"
027       END
028     END
029   NEXT I
030 NEXT PASS
031 PRINT "END"
032 STOP
033 *
034 100 * DECODE
035 BEGIN CASE
036 CASE (HEXCODE="03") OR ("20"=HEXCODE) OR (HEXCODE="22")
037   PTR = PTR+4 ; LOC = LOC+2
038 CASE HEXCODE="24"
039   PTR = PTR+4 ; LOC = LOC+2
040   GOSUB 200 ; * OUTPUT STATEMENT
041 CASE (HEXCODE="05") OR (HEXCODE="55")
042   LOOP
043   NEXT.BYTE = INSTRUCTS[PTR, 1]
044   PTR = PTR+1
045   UNTIL NEXT.BYTE = CHAR(254) DO REPEAT
046     LOC = LOC+6
047 CASE HEXCODE = "06"
048   HEX.STRING = INSTRUCTS[PTR, 4]
049   PTR = PTR+4 ; LOC = LOC+2
050   GOSUB 300 ; * HEX TO DEC
051   IF DEC.VALUE > 4096 THEN DEC.VALUE = DEC.VALUE-65536
052   TARGET = LOC+DEC.VALUE-1
053   IF PASS = 1 THEN LABELS<-1> = TARGET ELSE
054     STMT = "GO TO ":TARGET: " "
055     GOSUB 250 ; * OUTPUT GOTO
056   END
057 CASE ("10"<=HEXCODE) AND (HEXCODE<="14")
058   PTR = PTR+4 ; LOC = LOC+2
059 CASE (HEXCODE="30") OR (HEXCODE="32") OR (HEXCODE="34")
060   PTR = PTR+12 ; LOC = LOC+6
061 CASE HEXCODE = "40"
062   LOOP
063   NEXT.BYTE = INSTRUCTS[PTR, 1]
064   PTR = PTR+1 ; LOC = LOC+1
065   UNTIL NEXT.BYTE = CHAR(254) DO REPEAT
066 CASE HEXCODE="76"
067   GOSUB 200 ; * OUTPUT STATEMENT
068 CASE (HEXCODE="77") OR (HEXCODE="7C")
069   GOSUB 200 ; * OUTPUT STATEMENT
070 CASE (HEXCODE="7D") OR (HEXCODE="7F")
071   GOSUB 200 ; * OUTPUT STATEMENT

```

## UNCOMPILER VERSION THREE



```

072 CASE HEXCODE="78"
073   GOSUB 200 ; * OUTPUT STATEMENT
074 CASE (HEXCODE="B1") OR (HEXCODE="B2")
075   LOOP WHILE OCONV(INSTRUCTS[PTR,1], "MX") # "C1" DO
076     PTR = PTR+5 ; LOC=LOC+3
077   REPEAT
078 CASE (HEXCODE="C1") OR (HEXCODE="C3")
079   GOSUB 200 ; * OUTPUT STATEMENT
080 CASE HEXCODE="01"
081 CASE 1
082   UNKNOWN = 1
083 END CASE
084 RETURN
085 *
086 200 * OUTPUT STATEMENT
087 IF PASS # 2 THEN RETURN
088 STMT = "STATEMENT"
089 250 *
090 IF ALREADY.PRINTED THEN PRINT " ; ":
091 REFERENCED = 1
092 LOCATE LABEL.NUM IN LABELS,1 SETTING IGNORE ELSE
093   REFERENCED = 0
094   END
095 IF REFERENCED THEN PRINT LABEL.NUM: " ":
096 PRINT STMT:
097 ALREADY.PRINTED = 1 ; LABEL.NUM = LOC
098 RETURN
099 *
100 300 * HEX TO DEC
101 DEC.VALUE = 0
102 FOR H = 1 TO 4
103   DIGIT = HEX.STRING[H,1]
104   IF DIGIT >= "A" THEN DIGIT = SEQ(DIGIT)-55
105   DEC.VALUE = DEC.VALUE + (DIGIT#PWR(16,4-H))
106 NEXT H
107 RETURN
108 *
109 END

```

## UNCOMPILER LISTING CONTINUED

Ⓟ

```

001 100 Y=2 ; 150 A=B
002 GO TO 300
003 C=D ; 200 GO TO 100
004 300 Z=5 ; GO TO 400
005 350 I=1 ; C=2
006 GO TO 200
007 400 Y=99
008 GO TO 350
009 END

```

```

46 STATEMENT ; STATEMENT
GO TO 77
STATEMENT ; 73 GO TO 46
77 STATEMENT ; GO TO 116
91 STATEMENT ; STATEMENT
GO TO 73
116 STATEMENT
GO TO 91
*
END

```

A sample nonsense program shown as original source code on the left, and regenerated source code output by LABEL.GEN on the right. LABEL.GEN simply uses a statement's actual execution address for a label when needed; a program like the RENUMBER utility presented in *Pragma #1* can be used if "nice" labels are desired. Note how LABEL.GEN could not output a label equivalent to the original label 150, since 150 is not referenced anywhere by the program.

Ⓟ



# An Introduction to ENGLISH®

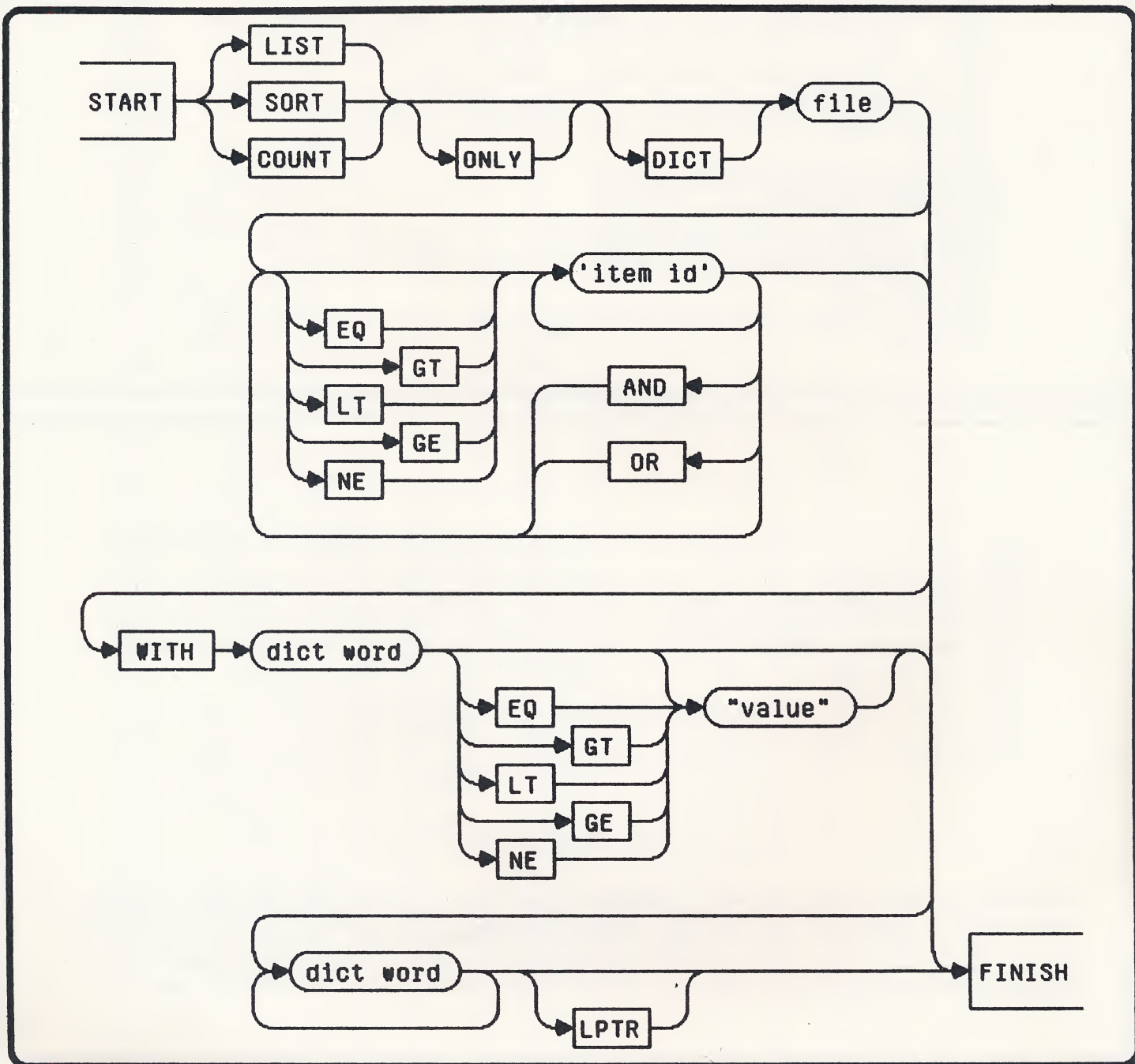
## Part 5: Syntax Overview

The fifth in a series of articles is presented for the beginning user of the inquiry and report generation language called ENGLISH. A syntax chart is introduced to aid in the creation of valid ENGLISH commands.

All of the various syntactic rules of ENGLISH that have been presented in the previous four installments of this series are contained in the chart shown below, which is designed to help you create correctly formed ENGLISH commands. To use the chart, START at the upper left and follow the arrows through the diagram until you reach FINISH. Numerous paths are possible. Each path represents a valid command. Upper case words in rectangles (such as SORT) must be typed exactly as shown, while lower case words in ovals (such as *file*) must be replaced with your own appropriate choice. For example, you might choose MD for *file*. Note that single quotes are required around *item id*, while double quotes are required around *value*.

Any paths that go through the WITH box form a type of command that has not yet been discussed. See if you can determine what an ENGLISH command does when a WITH clause is used. An example of such a command would be LIST CUSTOMERS WITH ZIP "95003".

Ⓟ





# OEMS-DEALERS SYSTEM INTEGRATORS

*A new day has dawned in the PICK™ marketplace*



**bantam computers, inc.**

## **ANNOUNCES THE BANTAM 002™**

If you would like to find out why we've got the best value in the Pick marketplace, register for one of our ISO Seminars to be held in November and December.

Seminars will be held in Chicago, New York, Washington, D.C., Dallas, Atlanta and Los Angeles.

You will hear about the best packaged, fastest, and best supported microcomputer utilizing Pick — available in both desktop and floor models, and based on the Motorola MC68000® chip. You will also hear about DB/OS which incorporates the Bantam SHELL™ and provides individual users with an advanced personal workstation.

To register and receive preliminary information, call or write:

### **ISO REGISTRATION**

**bantam computers, inc.**

**6033 W. Century Bl., Suite 1200**

**Los Angeles, CA 90045**

**(213) 642-4900**



# producer profile

Sunland, California is the new home of AccuSoft Enterprises, a software vendor that has specialized in the business graphics field of the Pick operating system. For this issue's profile, Pragma interviewed brothers Pete and Mike Schellenbach, owners of AccuSoft.

**Pragma:** ACCU-PLOT II is your one and only product. What does it do?

**Mike:** ACCU-PLOT II is a business graphics software package which is an extension to the inquiry processor on the Pick operating system. Our product produces line, bar, scatter and pie charts. It uses standard ENGLISH® syntax and interfaces with existing data files on the Pick system, so there is no need to create special files or to learn a complicated instruction set.

**Pragma:** Does your package have any competition?

**Mike:** We do have one competitor down in San Diego, who has a look-alike package which also uses an ENGLISH interface. The difference between their package and ours is that ours offers the ability to interface with various output devices, as opposed to working strictly with a Printronix printer. We also have the ability to generate pie charts through ENGLISH as opposed to writing pie charts with a BASIC program, which is what our competition is currently doing.

**Pragma:** If graphics output is such a great thing and there are so few of these products out on the market, how come you two are not retired millionaires?

**Mike:** I don't think a lot of people right now are interested in graphics or, if they are interested, they're waiting to see where the graphics industry is going and if there are going to be cheaper products on the market.

How are hardware and software products created? How are they sold and marketed? Why are some successful and others are not? What should be done to make the next product even better?

Regardless of the size of a company that is selling some product, often only one or two employees at the company will control the engineering and marketing of the item being sold. In this regular department, Pragma will be interviewing those personnel intimately involved with the creation of popular and well known products in order to reveal the who, what, when, where, why and how of organizations currently offering goods and services to Pragma readers.

**Pragma:** You're saying cost is the main problem to overcome in selling graphics products?

**Mike:** I think cost has a lot to do with it. I also think people are tentative about the claims we make with our product — whether or not it will do the things we say it will. The only way someone will really become interested and excited about a graphics package is if they have a chance to use it and see the output from their system.

**Pragma:** What's an example of something users would have a hard time believing your package can really do?

**Mike:** I think the biggest problem we have to overcome is the part about our package being an extension to the ENGLISH processor. They don't believe it's easy to create a graph by typing an ENGLISH statement which interacts with your data and outputs a graphical representation as opposed to a listing.

**Pragma:** How do you overcome these difficulties in selling?

**Mike:** Usually the way I would approach that is to show someone a listing of their data files that would be easily represented on one page of a graph as opposed to, say, twenty-five pages of a listing. That is a very good demonstration for people.

**Pragma:** That sounds like a very personal approach to selling a product. How do you actually manage to do it? You can't fly to every site in the country.

**Mike:** We manage to do it through our sample graphs which we send out to people. We send out sample listings of data and include sample graphs from that data.

**Pragma:** Some software vendors advertise prices and some don't. AccuSoft has always been in the "don't" category. What are the pros and cons of attaching a price to your software when you advertise?

**Mike:** When we first started advertising in *Pragma*, it was our policy not to include the price with our advertising. We have changed that, and we now include the price of our package on all of our advertisements. I feel it's important that people see the price, that they know exactly how much it's going to cost them for the package, and also that they can compare the price of our package to our competition very easily.

**Pragma:** What caused you to change your advertising policy?

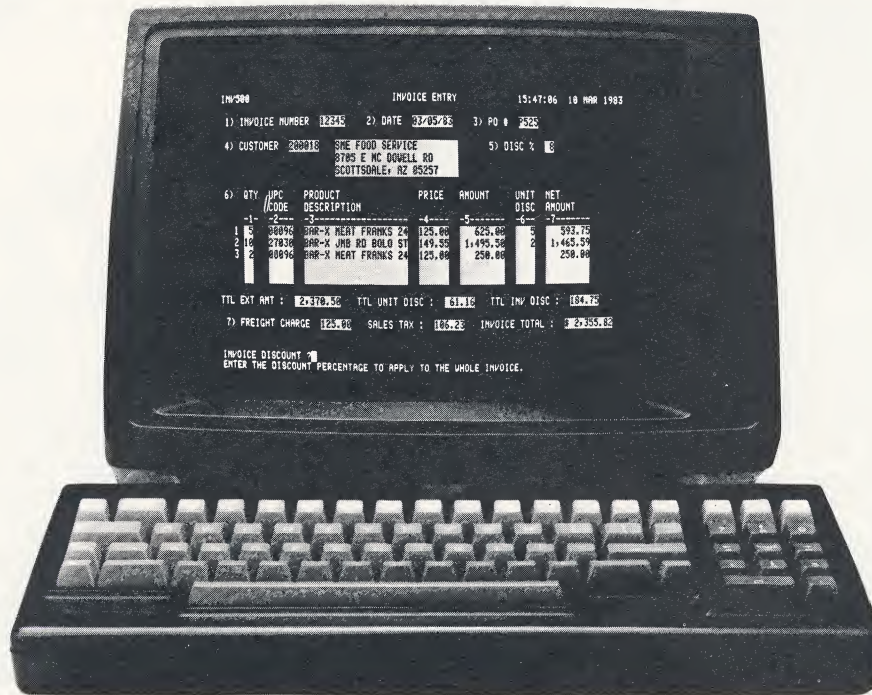
**Mike:** I think the biggest single reason was our competition charges 50% more for their product than ours.

**Pragma:** How did you determine the cost of your product?



NEW FROM INTERACTIVE SYSTEMS

# SCREEN-GEN



## What is SCREEN-GEN?

SCREEN-GEN is a data entry or data inquiry processor that can be used to develop program applications such as: order entry, journal entry, inventory adjustment, customer inquiry, or any other screen-related process.

SCREEN-GEN has a language processor similar to ENGLISH or ACCESS that works with your data dictionaries in defining a screen program, its format, prompt and function fields, and the file update logic. Once generated, these programs can be run interpretively by the system, simply by keying in the program name.

SCREEN-GEN optionally allows you to generate a screen program into BASIC code. SCREEN-GEN generates comprehensive documentation on each program defined. It also includes a full screen editor for maintaining screen program definitions once they are created, and has a dictionary

editor for maintaining your data dictionaries.

SCREEN-GEN is easy to use, yet is comprehensive in its approach to solving data processing problems. Utilizing multi-valued data structures is no problem with SCREEN-GEN. You can define any number of windows in your screens that scroll forward or backward independently and allow full control of your multi-valued data.

With SCREEN-GEN, you can define programs that extend to multiple screens for data entry and update any number of files within your database. For customized processing, you can even include BASIC code statements and routines directly into your screen definitions that will allow you to accomplish in SCREEN-GEN what you might have achieved in a user-written program.

SCREEN-GEN is written entirely in BASIC; there are no assembly

language routines used and source

code is supplied to each user. You no longer need to be dependent on anyone else for the maintenance of your system. SCREEN-GEN is available for all PICK-compatible systems. **To learn more about SCREEN-GEN, call (602) 993-3579 or mail the form below to:**



**INTERACTIVE SYSTEMS**  
129 East Voltaire Avenue  
Phoenix, Arizona 85022

### I'm interested in learning more about SCREEN-GEN.

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_  
Phone ( ) \_\_\_\_\_

SCREEN-GEN is a trademark of Interactive Systems. ENGLISH is a trademark of Microdata Corporation, Irvine, CA. ACCESS, PICK and PICK Systems are trademarks of PICK Systems, Irvine, CA.

**SCREEN-GEN is to data entry what ENGLISH is to report generation.**



**Mike:** I think we had a list of numbers on a piece of paper and threw darts at them! Seriously, we took a look at the amount of code we have in our product, a look at the amount of code in the operating system (in order to determine what percentage our code represents of a total system), and then determined the dollar value of the Pick operating system itself and took a percentage of that and charged that for our product.

**Pragma:** Software today is primarily sold in one of three ways: direct from the vendor, by dealers, or bundled with a manufacturer's hardware. What are the good and bad points of those three approaches for you?

**Mike:** Let's start off with bundled. I think the good point would be having a product out with every machine shipped. It increases exposure to the marketplace. The benefits of selling through a dealer would be that we can hit a variety of local users who would be unattainable through our usual operation. The dealers can go out and demonstrate the product to their clients. It's a little bit easier for the marketing of the product through dealers. The bad point about a dealership and bundling are that the prices are reduced and the net income for the business is a little bit less. However, we expect to sell more products with a reduced price, which will even out in the long run. Direct sales is a little bit harder of an approach as you have to do most of the sales through mailouts, and it decreases the amount of time you have to spend with a customer.

**Pragma:** Of those three categories, how is the product primarily being sold now, and how would you prefer it to be sold?

**Mike:** Right now I believe we've sold approximately 130 to 135 copies of our program. 90% of those have been through direct mail and advertising we've done in various publications. 10% of our sales have been through dealers and vendors throughout the country. We also have a large number of systems out there on the General Automation Zebra box.

**Pragma:** How is the Pick marketplace developing compared to some of the other operating system markets out there?

**Mike:** I see the Pick world expanding and growing at a very good rate, but I see other operating systems and other machines on the market that are growing faster than the Pick world right now. I think the emphasis Pick has to take would be to go into the IBM PC market and expand into low cost models, so the product would grow at a faster rate.

**Pragma:** Has the success of other operating systems affected your long range marketing plans?

**Mike:** Our background has been in the Pick world, and we're firm believers in the Pick operating system. I don't think we have any intention of going to other operating systems at this point.

**Pragma:** Why did you choose to write your product in assembler, and how does the current proliferation of Pick implementations affect you because your product is 100% assembly language?

**Pete:** The reason it's 100% assembly language is twofold. First, speed. When you're dealing with non-intelligent raster type output devices, you have to build a picture in memory before you can send to a device, and doing that means manipulating a great deal of disk space. The only way to do that efficiently was through assembler. The second reason is the linkage to ENGLISH. There is no efficient way to take data from ENGLISH, do something with it, and return back to ENGLISH for the next piece of information. So to keep the thing truly ENGLISH compatible required us to put it in assembler language. The problem with all the new machines is a little frustrating. It gives us a lot of work keeping up with

and doing the translations from machine to machine. It's not as bad, after doing two or three, as when we first started, but it's much more difficult than BASIC would be, just having to recompile.

**Pragma:** How many different versions of your product are there, because of the various different Pick implementations?

**Pete:** Seven.

**Pragma:** How many years have you been in this business?

**Pete:** We started development on the graphics package around 1978 to 1979. We actually had a version that worked around the end of 1979.

**Pragma:** Was the product originally developed for your own personal needs, and then you decided to market it?

**Mike:** I think the original reason the package was developed was to exercise the inherent graphics capabilities of the Printronix printer, and to offer some kind of in-house graphics for managerial evaluation.

**Pragma:** How have you seen the Pick market evolve during the past four years?

**Pete:** I think the market hasn't expanded at the rate which we anticipated it would. When we first started working on this, basically there were two Pick systems you could buy. There was one sold by Microdata, and there was one sold by Pick and Associates, who sold it to Evolution. Things started looking a little bit more encouraging when Ultimate came along. Then ADDS arrived and we thought "boy, we'll have a nice cheap system and we'll see thousands and thousands of them out there on the market", but that really hasn't happened yet. It's looking better and better. There are more and more vendors picking up the line.

**Pragma:** How has the operating system been good for your particular product?

**Pete:** The operating system does a heck of a lot of work for us. It's very difficult to move this product to another operating system because you don't have ENGLISH. The driving force behind our product was that we had this wonderful report generator and database management system that could go and do super things, and all we had to do was tell it to make a picture instead of a listing. The greatest advantage of the Pick system, I think, is the way it handles the data. In most operating systems, a database system and a report generator and an inquiry language are all optional items. You have to buy them from third party vendors and they are not built into the operating system.

**Pragma:** How would you like to improve the inquiry language?

**Pete:** I think there are a lot of enhancements that could be made. Some of the enhancements that have been made over the past few years, although there have not been many, have been useful, such as being able to use an alternative dictionary instead of the standard dictionary, and being able to use A-correlatives instead of F-correlatives, which were hard to understand. If the system could document itself in a little bit clearer way, it would make things quite a bit easier for the user. Also, there are some capabilities, like taking the first ten records out of a file instead of having to use some other kind of selection criteria, that would be useful. I don't see any reason why they can't be done, except that somebody doesn't have time to write them.

**Pragma:** What's a more specific example of what you mean by the system documenting itself?

**Pete:** Having some kind of utility to keep track of what dictionary items are used in what applications, with descriptive



# AUROPLAN<sup>TM</sup>

**AUROTECH**  
OF COLORADO

## **Now Software Similar To VisiCalc<sup>®</sup> Is Available On The Pick Operating System**

**A financial planning and modeling system that surpasses all others. Ideal for preparing financial models, budgets, cost estimates, sales recaps, scheduling and resource reports and analyses.**

**AUROPLAN** is an electronic worksheet designed to drastically reduce the time and effort required to prepare financial reports.

**AUROPLAN** has a trial value feature which allows you to ask "What If?" without destroying previously entered data or formulas.

**AUROPLAN** allows you to name elements in one worksheet and reference them in another, so you can construct modular worksheets and link them together. This eliminates the need for huge worksheets and the long calculation times associated with them.

**AUROPLAN** can retrieve data directly from your existing

computer data base without the need for costly interface programming.

**AUROPLAN** can be learned in just a few hours. The only limit to its capability is your imagination.

That's **AUROPLAN!** Now available for use with the Pick Operating System.

For more information on  
**AUROPLAN**, contact:

**AUROTECH**  
OF COLORADO

925 S. Niagara  
Denver, CO 80224  
(303) 388-1612

VisiCalc<sup>®</sup> is a trademark of VisiCorp, Inc.



names, rather than saying LIST DICT on a file and seeing some name like ACC with three periods after it, and having no idea what it's for. If there was a way to say "this is a three digit account number to choose for the general ledger", it might make using the system quite a bit easier.

**Pragma:** In the microcomputer market, graphics software tends to be very interactive, integrated very closely with other application systems, and fairly inexpensive. The user can display a graph on the screen and make changes without having to bother with ENGLISH-like commands. How do you see yourselves competing with that?

**Mike:** We looked at a number of the graphics applications available on the micro level, and we realize there is definitely a need for our product to become more user oriented so they can make the necessary adjustments to the graph. Right now, that capability is not supported with our product, but we hope to have a third generation graphics package available sometime in late 1984 that will be more user integrated.

**Pete:** We've found most of these inexpensive graphics systems in the micro market do not have sufficient resolution to do the kind of graphics we do. Most of them are on the order of the IBM PC which, I think, is something like 480 horizontal by 200 vertical positions. It is the vertical resolution that is usually a problem on them.

**Pragma:** If you were to buy a Pick system today, which model would you buy and why?

**Mike:** You want us to get in trouble with the manufacturers?

**Pragma:** Let's say you have made your choice. Without naming who that would be, what is it about the system that sold you?

**Pete:** One of the main things is speed, both doing development type work and in general processing. One of the other major points is price. There is such a wide range in prices for systems out there, from hundreds of thousands of dollars to tens of thousands. You have to arrive at a good price/performance ratio.

**Pragma:** If you computed some sort of price/performance ratio for each machine and then just picked the "optimum", are you saying that would be a good enough way to make a choice and other aspects wouldn't be significant?

**Pete:** Some of the other aspects are still important. One of the other things that is very important is whether it's going to do the job you want it to do. If all the software executes properly. If the system is solid. I think the only way you can judge that is on a case by case basis, depending on the mix of software you want to put on.

**Pragma:** What about a company that exclusively sells Pick systems, such as Microdata or Ultimate, compared to a company that deals with many other products, such as Altos and Prime? Can the diversified companies really offer quality Pick products and support while also concentrating on unrelated product lines?

**Mike:** I don't know if that has anything to do with the kind of support you would get from a dealer or manufacturer. Many of them are using outside field service. Most of them have some kind of technical support. I feel a lot of companies, when they first introduce the Pick operating system as a product in their line, have a certain learning curve their personnel are going to have to go through before they are familiar enough with the system to offer the kind of support the user really needs. Once they reach a point in the learning curve where they are familiar with the system and can handle the problems that arise, there should not be a difference between the service offered by that company and a company that deals strictly with Pick.

**Pragma:** What is the thing making it most difficult for AccuSoft to become the company it wants to be?

**Mike:** I think basically we don't have enough hours in the day, and there is a shortage of experienced personnel to handle the type of development we want to do. I think eventually we will have to hire at least one other systems analyst to help with the development of our third generation product.

**Pragma:** There has been a lot of talk about Pick's upcoming new release. What will that mean for you?

**Pete:** We hope it will enable us to continue the development on our products a little bit more easily. There are so many restrictions when we're working at the assembler level of the system right now. We hope some of those will be eased. On the other hand, there may be some significant changes in the system which will cause great headaches.

**Pragma:** Does that mean you're just going to have to wait like everyone else to see what the new release is and then adjust to it as best you can?

**Pete:** Hopefully with a tiny bit of warning.

**Pragma:** What does the Pick system lack and what would you like to see the new release provide?

**Mike:** I think the biggest drawback to the Pick operating system now is twofold. First of all, there is a lack of communication. I'd like to see the ability to tie together various Pick machines so they can share databases. The second drawback to the Pick operating system now is it does not support high speed languages other than assembler. I'd like to see Pick support something in the line of the C programming language.

**Pragma:** Did you make any attempt at all to use BASIC in your product, perhaps isolating what the real bottlenecks were and only doing those portions in assembly language? Or did you essentially assume the whole product would have to be in assembly language?

**Pete:** When we were first doing the analysis for the development of the package, we wrote a great deal of it in BASIC and found the speed was completely unacceptable. So, we started developing the backbone of the package in assembly language, with the idea that part of it could still be in BASIC, but when it came to tying it into the ENGLISH processor, there was no way to keep part of it in BASIC and still have it running.

**Pragma:** What else in the Pick operating system needs to be improved?

**Mike:** One of the things I've had complaints about from other users is that it's not as easy to use as it ought to be. It would be very helpful to come up with some teaching tools to simplify the use of the system for inexperienced users. Some of the manufacturers have now started doing that with something like the dot processor on the Prime and on the ADDS. Things along that line are most useful.

**Pragma:** What things do you feel are particularly nice about the Pick system, that are going to give it an advantage over other operating systems for some time?

**Mike:** Mainly the filing system. The fact that a file can have any number of records, and any record can be virtually any size and have any number of fields and have any number of subfields in those fields, and that the length of any of those things is irrelevant. The fact that the system can handle that without any problems at all and still run at an acceptable speed is incredible.

P



**Q:** Why should I let  
**WIZARD** create the  
computer programs  
I need?

**A:** "We were asked by our Marketing Services department to develop a Bill of Materials and Material Requirements plan for their new product line. Using **WIZARD** and **ENGLISH**®, within four hours we had designed, programmed and entered the BOM file, and produced the required reports. You can imagine the reaction when our response to previous requests had either been *NO* or *give us two months and a few thousand dollars.*"  
— G. W. Gersbach, Controller, Lister Australia

**A:** "How else could I produce a bug free, fully documented input program in half an hour? **WIZARD** is to data entry as **ENGLISH** is to data output."  
— Bob Guthrie, Owner, Guthrie & Associates

**A:** "We are a first-time user, and have found **WIZARD** file maintenance procedures to be easily understood by our staff, and much simpler than using the dealer's package."  
— A. K. Cole, Chief Accountant, Permaglass

---

**Q:** How can I try **WIZARD**?

**A:** For a 30 day trial, send your check for \$100 to Automatic Programming. We will send you the entire family of **WIZARD** products on magnetic tape along with complete instructions.

Automatic Programming Inc. • 85 Lakeshore • Irvine CA 92714 • 714-552-2800

ENGLISH is a trademark of Microdata Corporation.



# benchmarks

Are you thinking of doing an upgrade to your hardware or software? Are you comparing throughput and performance while shopping around for a system? Have you converted from one machine to another? Be sure to send in your benchmark statistics to Pragma, so the results can be featured in this department and shared with other installations.

## Rounding Out 7 Benchmarks On 5 Machines

A new test of dynamic array speed, plus missing timings for certain machines from benchmarks previously published here, are gathered together in one chart to allow comparing all machines on all tests.

In previous issues, this department has, among other things, presented 16 different timings from a selection of six benchmark programs on four different computers: the IBM PC with Revelation, the Mentor, the Sequel™ and the Reality®. Not all benchmarks were run on all machines, especially the Sequel and Mentor.

In order to better show the relative speed of these systems, and to complete the collection of benchmarks, all missing figures were obtained by running all programs on all machines, generating 19 new timings. Also, a seventh test program (see the listing below) was written and included to test the speed of dynamic arrays. Finally, all seven benchmarks were also run on the Zebra, and the chart on the opposite page was created to present the results.

Our thanks to Mark Pick of General Automation for running the Zebra tests, to Steve Kowarsky of Rainbow Natural Foods for providing more Mentor 4060 timings, and to Roger Harpel of Cosmos for checking the PC. Any readers interested in running the benchmarks on their particular machine, if not shown here, should contact Pragma to order a free tape containing the test programs, so that we can publish the results in a future issue.

P

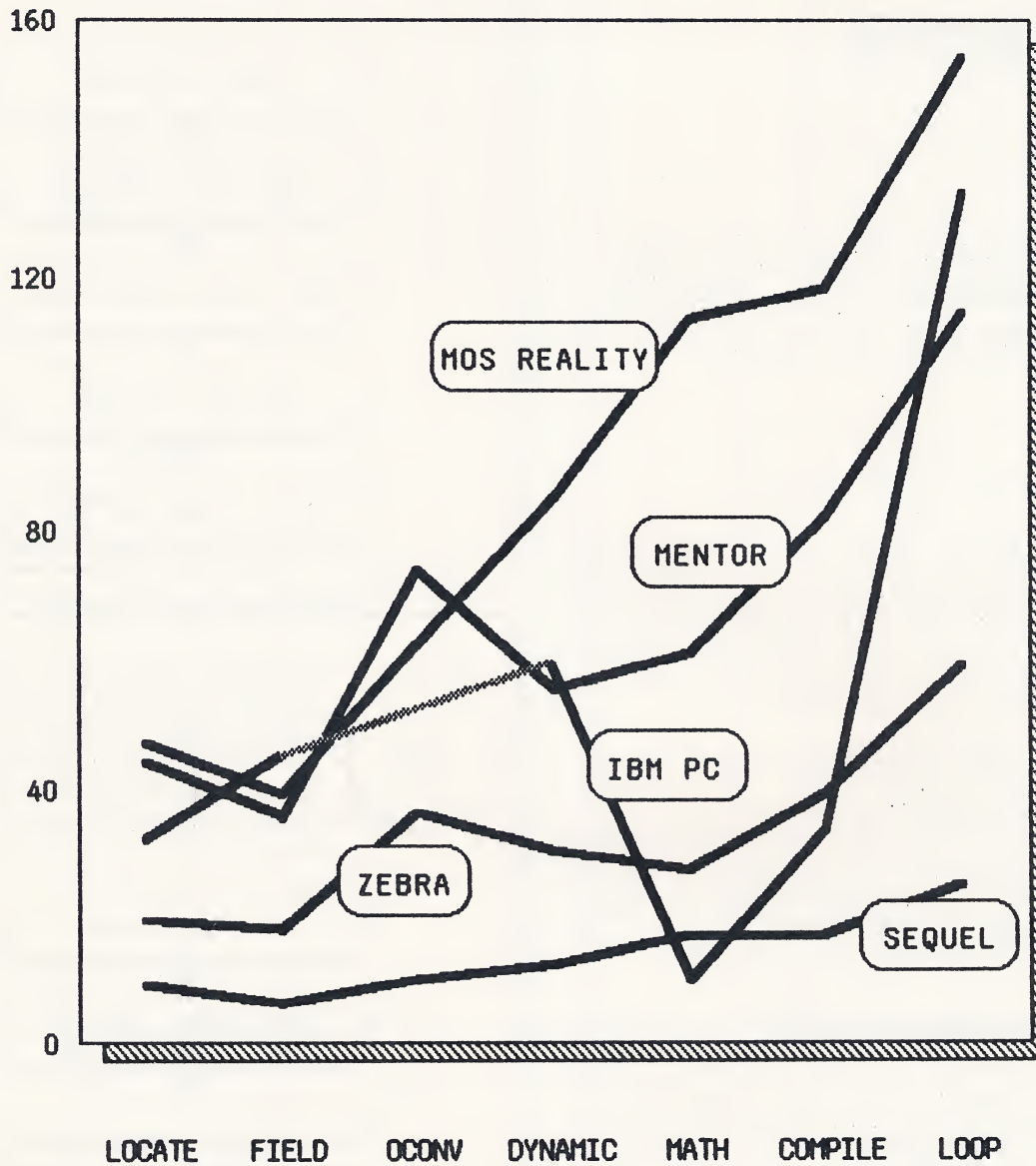
```
START = TIME()
X = ""
FOR I = 1 TO 50
  X = INSERT(X,I,0,0,I:I)
NEXT I
LOOP
  SWAPPING = 0
  FOR I = 1 TO 49
    IF EXTRACT(X,I,0,0) < EXTRACT(X,I+1,0,0) THEN
      SWAPPING = 1
      TEMP = EXTRACT(X,I+1,0,0)
      X = REPLACE(X,I+1,0,0,EXTRACT(X,I,0,0))
      X = REPLACE(X,I,0,0,TEMP)
    END
  NEXT I
  WHILE SWAPPING DO REPEAT
    FOR I = 1 TO 50
      X = DELETE(X,I,1,1)
    NEXT I
  PRINT (TIME()-START):" SECONDS"
  STOP
END
```

**DYNAMIC  
BENCHMARK**



## 34 Benchmark Timings for 5 Machines

TIME, in seconds



(Note: the IBM PC OCONV time was off scale at 610 seconds.)



# Designing Data Entry Programs

Consistency of design and flow of control in data entry programs is discussed. A set of increasingly detailed charts is presented to give an overview of different types of data entry requirements and to act as an aid in designing data entry programs.

A data processing system may contain a large number of data entry programs designed to accept user input and update file records. If all of the data entry programs in a system are consistent in design and style, then all of the programs tend to look similar. For both the programmer and the user, such sets of similar programs are easy to learn and use, because all programs will look and operate so much alike.

In the same way that report generating languages take advantage of the fact that a large number of different reports for a variety of applications can all be based on similar principles (such as sorting, using columns, and totaling fields), one consistent input system can be designed to handle the input requirements of many different types of applications. Unfortunately, data processing systems often consist of a mixed bag of data entry programs, with very little similarity in design and operation from one program to the next.

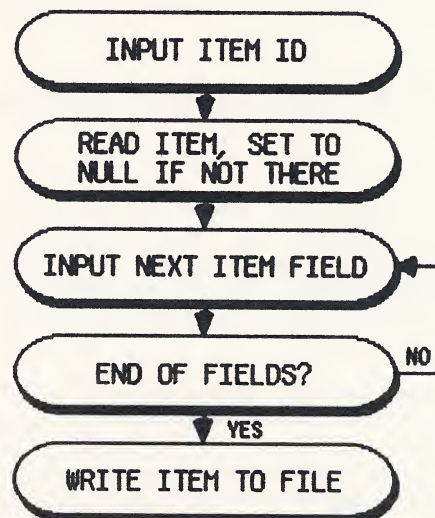
This presentation is an overview of various styles and levels of complexity that can be incorporated into typical data entry programs. The idea presented is that with a set of charts such as those shown with this article, the system designer can more easily select the best solution for a given application's data entry needs, while encouraging consistent design, implementation and operation in the final program.

The basic procedure an operator follows when using a data entry program is to request a file item (such as an employee's master record) by some unique item identifier (the employee's number), to edit the fields (name, address, social security number and so on) in the item, and to then store the item back in the file it came from.

What follows is a set of charts that begins with the basic procedure just described, and then repeatedly enhances it, resulting in a comprehensive, detailed procedure for a general purpose data entry program.

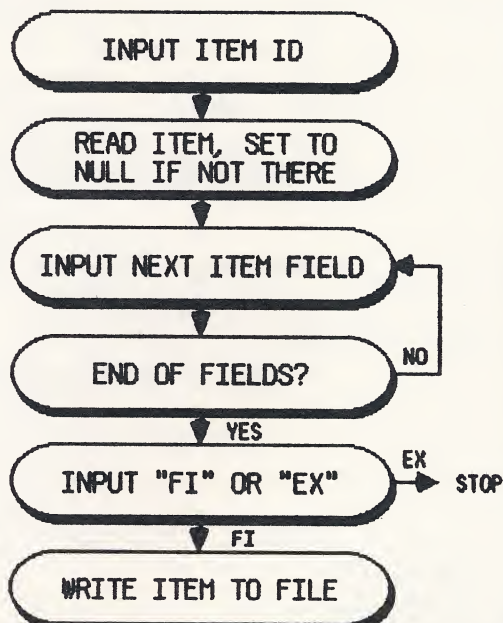
1

The simplest type of data entry program is one that merely steps the user through each field. Sometimes the application requires that the item already exist before the user may change it, but usually it is allowable to start with null fields if the item doesn't exist, and let the same program be used for creating or changing items.



2

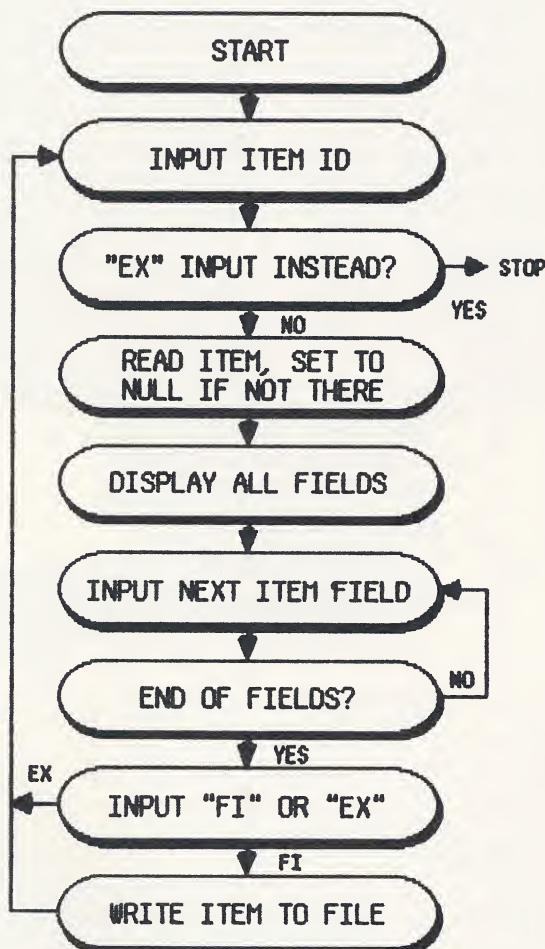
Generally, only certain applications or items with a very small number of fields can be automatically written back to the file. Instead, a final operator prompt giving the choice of filing the item or exiting from the program allows better control.





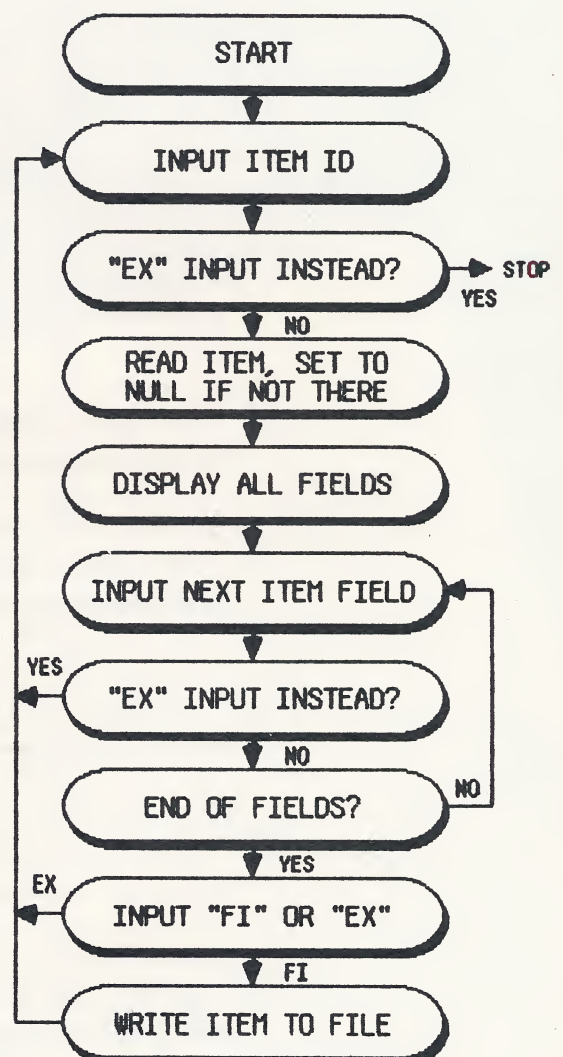
3

Since more than one item is often edited, it is usually convenient to have the program reprompt for another item identifier after the previous item is filed or if an exit is taken. Since this causes the program to stay in a loop asking for items, the program should recognize an exit at the item identifier prompt, allowing the program to stop. Also, it is helpful to the user to display existing data in all fields once an item has been called up from the file. (Displaying all data before any of it is edited is only practical at high terminal baud rates. But since systems running at low baud rates are configured more out of consideration for the hardware than for the users, those kinds of programs will be ignored here.)



4

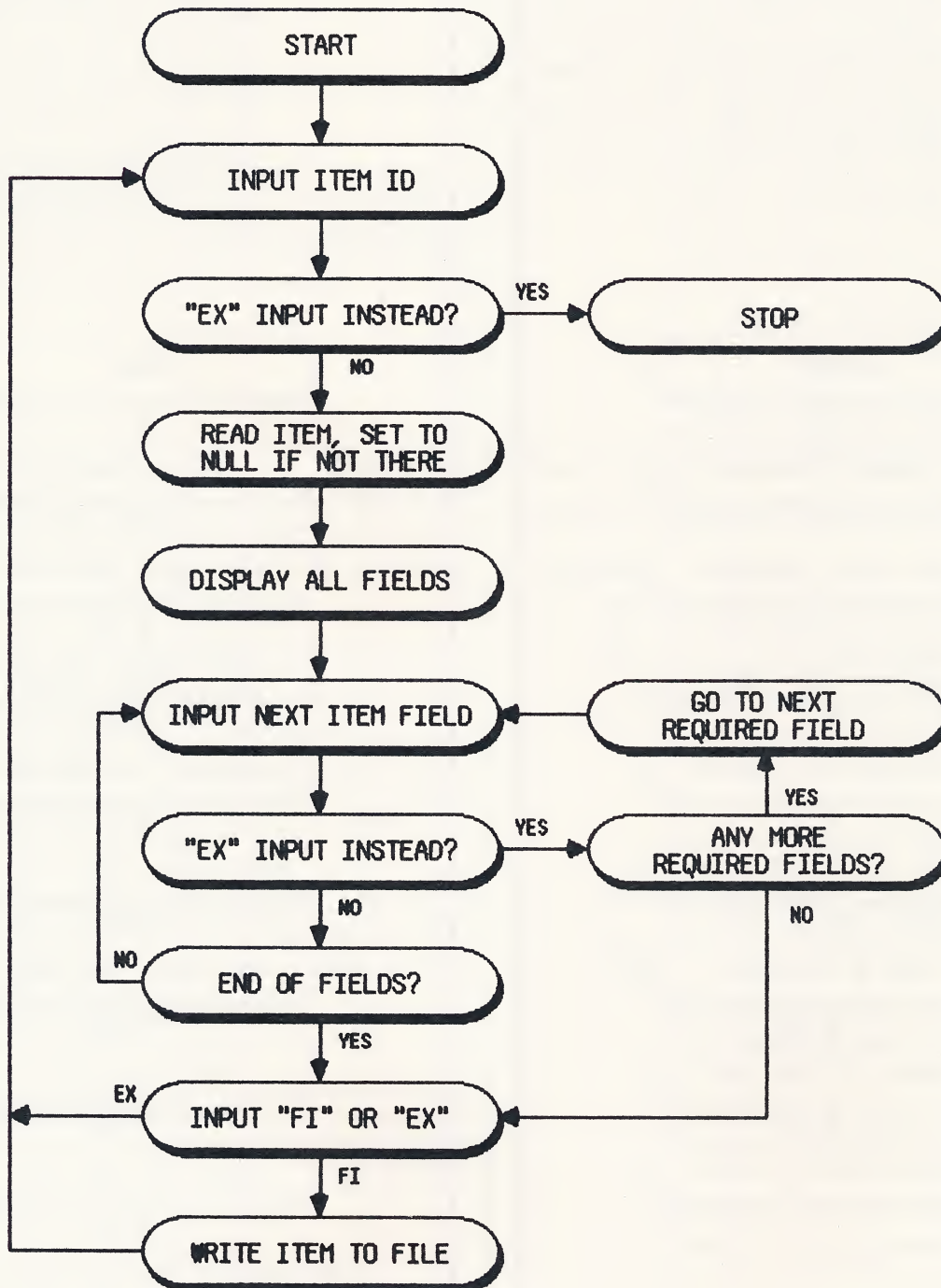
Very often, the item consists of a large number of fields, and the user ends up wanting to skip the remaining fields, or wanting to exit out of the item completely from some point in the middle of the item because of gross errors. Being able to exit at any field is very convenient, but where to jump when the user types EX in the middle of an item can make a big difference in the usefulness of the program. Jumping back to the initial item identifier prompt is easiest to implement, but means that the user has to step through all fields in order to get to the final FI/EX prompt (although that might not be so bad). If an EX in the middle of an item returns to the first prompt to start over, then we have the situation diagrammed here.





5

Jumping to the final FI/EX prompt when EX is used in the middle of an item can be simple too, but often fields in the item are required to be filled in, and allowing them to be skipped causes complications. Jumping to the next required field if EX is input is a possible, but complicated, solution.





# COMPU-SHEET™

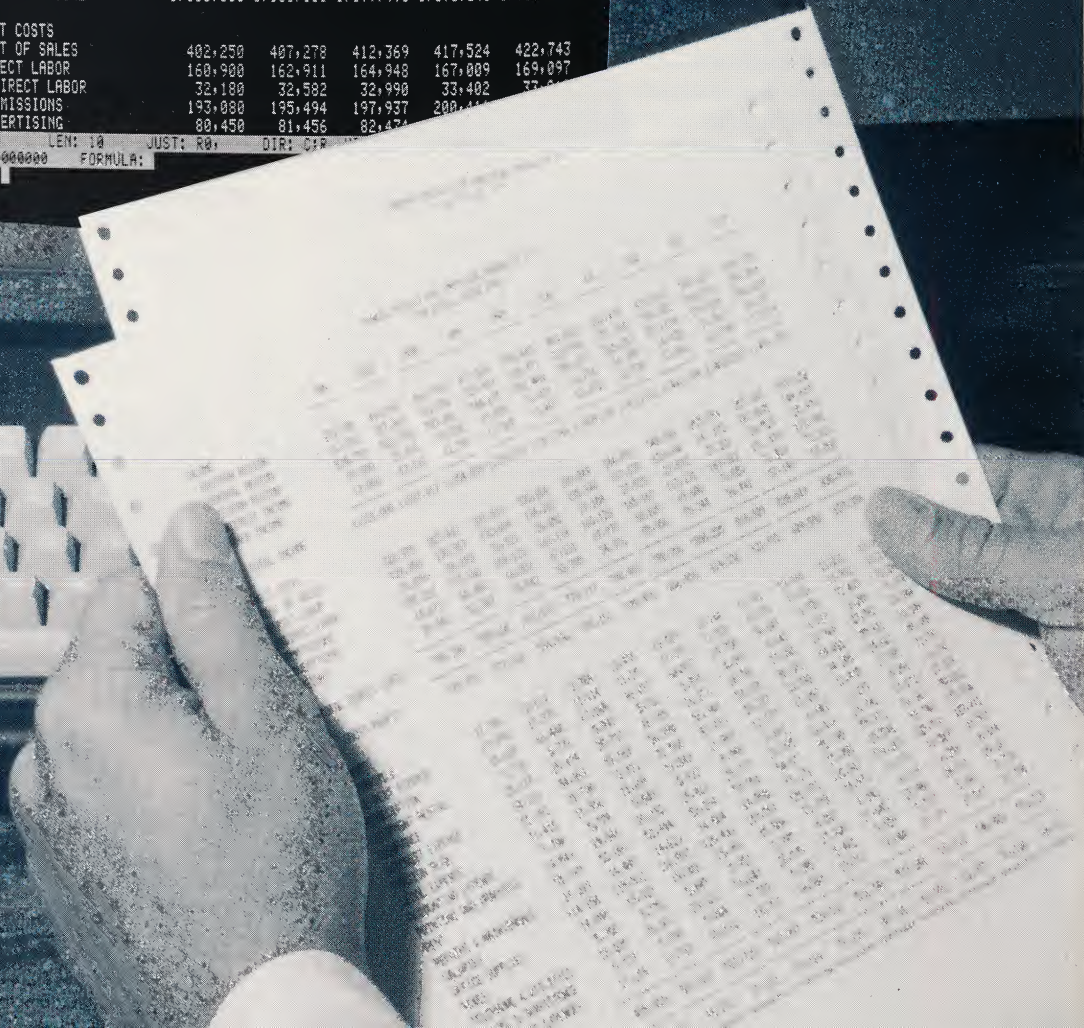
The electronic spreadsheet  
for your PICK™ Operating  
System computer that will  
save you hours of frustrating  
manual calculations.



ANNUAL BUDGET FOR AMERICAN PRODUCTS CO.  
FOR FISCAL YEAR 1983

|                 | JAN       | FEB       | MAR       | APR       | MAY       |
|-----------------|-----------|-----------|-----------|-----------|-----------|
| INCOME          |           |           |           |           |           |
| EASTERN REGION  | 543,000   | 549,788   | 556,660   | 563,618   | 570,663   |
| CENTRAL REGION  | 456,000   | 461,700   | 467,471   | 473,315   | 479,231   |
| WESTERN REGION  | 610,000   | 617,625   | 625,345   | 633,162   | 641,077   |
| INTEREST INCOME | 15,500    | 15,629    | 15,758    | 15,889    | 16,021    |
| OTHER INCOME    | 12,000    | 12,120    | 12,241    | 12,364    | 12,487    |
| TOTAL INCOME    | 1,636,500 | 1,656,861 | 1,677,476 | 1,698,348 | 1,719,479 |
| DIRECT COSTS    |           |           |           |           |           |
| COST OF SALES   | 402,250   | 407,278   | 412,369   | 417,524   | 422,743   |
| DIRECT LABOR    | 160,900   | 162,911   | 164,948   | 167,009   | 169,097   |
| INDIRECT LABOR  | 32,180    | 32,582    | 32,990    | 33,402    | 33,814    |
| COMMISSIONS     | 193,080   | 195,494   | 197,937   | 200,411   | 202,904   |
| ADVERTISING     | 80,450    | 81,456    | 82,471    | 83,496    | 84,531    |

LOC: 2.10 LEN: 10 JUST: R0 DIR: C/P  
DATA: 155000000 FORMULA:  
COMMAND:





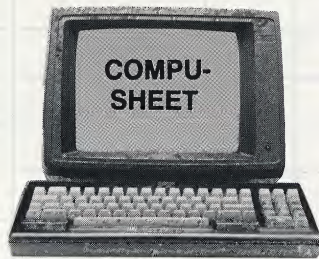
# What is COMPU-SHEET?

## Recently, electronic spreadsheet programs

have become an essential management tool for company presidents, CPA's, treasurers, controllers, managers or anyone who needs an easier way to solve problems! Programs such as VISI-CALC®, MULTI-PLAN™ and others have become among the most successful software products ever developed.

**There is good reason** for the success of these programs. First, they allow the user to simulate everyday business planning problems on their computer in a form that is both easy to use and easy to understand. Second, these tools allow anyone to utilize the computer for complex problem-solving without the technical expertise that was previously required. Now people without programming experience can design, enter and solve in minutes, problems which would take them days or weeks to complete using a pencil and calculator as they have done in the past.

**Now, Interactive Systems has developed COMPU-SHEET**, an electronic spreadsheet that offers the features of those powerful tools, PLUS the advantage of interfacing with the powerful PICK Operating System. COMPU-SHEET is specifically designed for non-programmers (although experienced programmers will find themselves utilizing COMPU-SHEET too). There is no reason to purchase a separate computer to run your electronic spreadsheet—COMPU-SHEET will run on your existing hardware and can be shared by all users on your system.



**COMPU-SHEET gives you**, in effect, a large blank sheet of "paper" where you can define and solve your problem. This sheet is divided into columns and rows. You define the number of columns and rows, the width of each column, and the format of the data (such as: whether the information is to be left or right justified, how many decimal points should be displayed, should dollar signs and credit symbols be displayed, etc. . .). At any time you may insert or add new columns or rows, or you may change the width or format of any column.

VISI-CALC® is a registered trademark of VisiCorp. MULTI-PLAN is a trademark of Microsoft Corporation. COMPU-SHEET is a trademark of Interactive Systems. PICK and PICK Systems are trademarks of PICK Systems, Irvine, CA.

**At any location** (the point where a column and row intersect) you may begin to define your problem. A location may contain one of several types of entries.

**First**, it may contain "heading" information. These are entries such as worksheet or column headings ("ANALYSIS OF LEASE VS. PURCHASE" or "SALES," "BUDGET," "VARIANCE," etc. . .). A large heading may spread across several columns.

**Second**, a location may contain any type of "data." This is virtually any type of information you may choose to enter. It may be a department name, product description, budget amount, sales amount, percentage, or any other type of alphabetic or numeric information you might need. The information entered here may be used in calculations elsewhere on the worksheet.

**Third**, a location may contain a formula. This formula is written in a simple algebraic format. The formula may operate upon any combination of locations, may contain any type of "constant" values, may retrieve information from any file in your data base (such as your general ledger files, sales files, financial reporting files, etc.), or may retrieve information from other worksheets created by COMPU-SHEET. You may choose to merge information from several worksheets into one "consolidated" worksheet. The power of COMPU-SHEET is limited only by your imagination.

| LIST SOURCE | QTY     | MAILED | RATE | RESP | AVERAGE | SALE  |
|-------------|---------|--------|------|------|---------|-------|
| LIST #1     | 45,000  | 2,116  |      |      |         | 43.55 |
| LIST #2     | 15,000  | 1,394  |      |      |         | 48.66 |
| LIST #3     | 25,000  | 1,637  |      |      |         | 33.74 |
| LIST #3A    | 17,500  | 1,313  |      |      |         | 34.56 |
| LIST #4     | 30,000  | 1,750  |      |      |         | 41.77 |
| PBC LIST    | 30,000  | 1,055  |      |      |         | 38.42 |
| PVE LIST    | 27,000  | 1,267  |      |      |         | 44.08 |
|             | 192,000 |        |      |      |         | 40.63 |

**WHAT IF?**

|               | JUN       | JUL       | AUG       | SEP       | OCT       | NOV       | DEC       |
|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| RESIDUAL      | 542,000   | 542,000   | 542,000   | 542,000   | 542,000   | 542,000   | 542,000   |
| COST OF SALES | 250,000   | 250,000   | 250,000   | 250,000   | 250,000   | 250,000   | 250,000   |
| GROSS PROFIT  | 292,000   | 292,000   | 292,000   | 292,000   | 292,000   | 292,000   | 292,000   |
| TOTAL INCOME  | 1,070,000 | 1,070,000 | 1,070,000 | 1,070,000 | 1,070,000 | 1,070,000 | 1,070,000 |

**Once your worksheet has been created**, you can start analyzing in ways which were never

available before. **WHAT IF** the interest rate is 11.5%?, 13%?, 15%? . . . **WHAT IF** sales increase by 18%, materials by 14% and payroll by 12% . . . **WHAT IF** the response rate is 1.5%, 2.0%, 2.5% . . . **WHAT IF** . . . . . ???

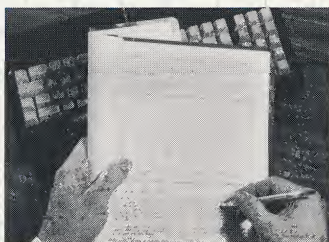
**COMPU-SHEET is truly a MANAGER'S tool.** It allows the manager to take advantage of the computer in the important functions of **PLANNING** and **CONTROLLING**.



## Is COMPU-SHEET easy to learn?

**Yes. COMPU-SHEET was designed for non-programmers.** There is a detailed reference manual which will step you through each available operation. In just a few hours you will be building meaningful worksheets.

**COMPU-SHEET utilizes English-like commands,** not cryptic and difficult-to-learn command codes. The computer prompts you each step of the way. But, if you run into trouble, just enter a "?" and COMPU-SHEET will display an explanation of what is needed. COMPU-SHEET is easy to master, easy to use, yet powerful.

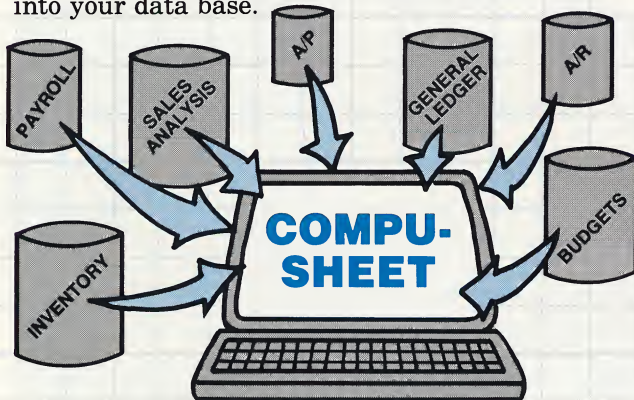


## Can COMPU-SHEET handle my problems?

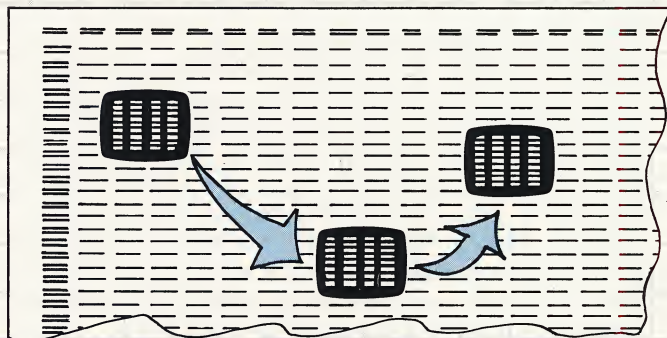
**COMPU-SHEET places few limitations on you.**

There is no limit to the number of columns or rows you may use. Worksheets may be linked together where information is transferred between any number of separate worksheets. COMPU-SHEET is appropriate for any size or type of business operation.

**Perhaps one of the most significant capabilities** of COMPU-SHEET is its ability to retrieve information from your data base. You can link information from anywhere in your system into your worksheets for very advanced analysis and projections. Powerful? Yes, yet it is very easy to do. Imagine the many ways your models can tie into your data base.



**One way to perceive COMPU-SHEET** is to view your screen as a "window" which displays part of a much larger worksheet. This window may be moved around the worksheet to "uncover" any



section you wish to see. You may divide your screen into several windows, each displaying a different part of the worksheet so you can enter or change data in one section and watch the results in other parts as the sheet is recalculated. Any of these windows may be "locked" in place while you scroll through others.

**When you're ready to print your worksheet** you will find COMPU-SHEET unmatched in its flexibility. You may ask to print the worksheet and COMPU-SHEET will automatically break it into as many pages as is necessary. You have the option to specify the size of each page. If you want, you can specify any combination of columns and rows to be printed (you can even specify that certain columns and rows are to be repeated on multiple pages).

**Formulas may be as simple** or as complex as you need to do the job. All standard algebraic operations are available plus a number of expanded operations. Even conditional statements (IF-THEN-ELSE) are supported and gives COMPU-SHEET the capability to handle complex relationships where advanced analysis is required.

## Where can I use COMPU-SHEET?

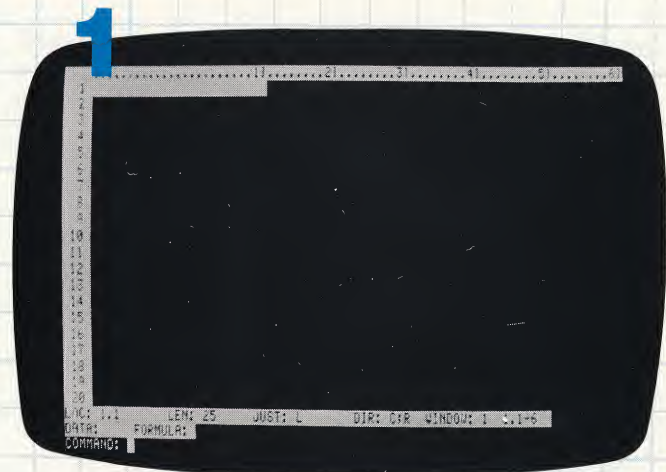
**COMPU-SHEET can be used for** cash flow analysis and projections ... Pro-forma statements ... Budget analysis ... Profitability analysis ... Job costing ... Financial statements ... Estimating ... Manpower assignments ... Sales forecasting ... Marketing analysis ... Vehicle operating costs ... Trust funds ... Proposals ... Labor and material requirements planning ... the list is endless!

**As you can see,** the uses of COMPU-SHEET are limited only by your imagination. Many everyday tasks can easily be handled by COMPU-SHEET. Most importantly, think of the analysis you will now be able to do ... analysis which really NEEDS to be done. With COMPU-SHEET, you now have the time and the means.



# How to use COMPU-SHEET

After specifying the worksheet size, COMPU-SHEET will display a blank worksheet. In this example, the first six columns (numbered at the top) and twenty rows (numbered on the left side) may only be part of a much larger worksheet. What you see here is a "window" on the worksheet. The window may be moved around so you can see any part of the worksheet. At the bottom of the screen are the "status" lines. These lines keep you informed of important information concerning current location size and format, window size, and any previously entered formulas. The last line is the "command" line where all commands are entered.



To begin building your worksheet, you simply start entering information. Here, we are entering the worksheet heading and column and row headings. A large heading (such as the worksheet name) may extend across several columns. Since the width of each column may vary, your worksheets may be designed to fit the unique requirements of each application. There is no restriction on the contents of any location. A location may contain a heading, any alphabetic or numeric information, or a formula which specifies any calculations which are to be performed to yield the needed results.



Here we have finished entering the data and formulas. COMPU-SHEET automatically calculates each formula as it is entered so you can test the worksheet as you are building it. Formulas may reference a location as either "absolute" (meaning a specific location) or "relative" (meaning a specific number of locations away from the location being calculated). By using this method of entering formulas, you may enter a formula once and then copy it to a range of other locations (this offers significant time savings in entering formulas). Any of these formulas may retrieve information from other worksheets or any file in your system.

|                 | JAN       | FEB       | MAR       | APR       | MAY       |
|-----------------|-----------|-----------|-----------|-----------|-----------|
| INCOME          |           |           |           |           |           |
| EASTERN REGION  | 543,000   | 549,788   | 556,660   | 563,618   | 570,663   |
| CENTRAL REGION  | 456,000   | 461,700   | 467,471   | 473,315   | 479,231   |
| WESTERN REGION  | 610,000   | 617,625   | 625,345   | 633,162   | 641,077   |
| INTEREST INCOME | 15,500    | 15,629    | 15,758    | 15,889    | 16,021    |
| OTHER INCOME    | 12,000    | 12,120    | 12,241    | 12,364    | 12,487    |
| TOTAL INCOME    | 1,636,500 | 1,656,861 | 1,677,476 | 1,698,348 | 1,719,479 |
| DIRECT COSTS    |           |           |           |           |           |
| COST OF SALES   | 402,250   | 407,278   | 412,369   | 417,524   | 422,743   |
| DIRECT LABOR    | 160,900   | 162,911   | 164,948   | 167,009   | 169,097   |
| INDIRECT LABOR  | 32,180    | 32,582    | 32,990    | 33,402    | 33,819    |
| COMMISSIONS     | 193,880   | 195,494   | 197,137   | 200,411   | 202,917   |
| ADVERTISING     | 80,450    | 81,456    | 82,474    | 83,505    | 84,549    |



# SHEET

Here is a step-by-step example of a typical COMPU-SHEET analysis. The basic procedure is as follows:

4

| ANNUAL BUDGET FOR AMERICAN PRODUCTS CO.<br>FOR FISCAL YEAR 1983 |                  |                  |                  |                  |                  |
|-----------------------------------------------------------------|------------------|------------------|------------------|------------------|------------------|
|                                                                 | JAN              | FEB              | MAR              | APR              | MAY              |
| <b>INCOME</b>                                                   |                  |                  |                  |                  |                  |
| EASTERN REGION                                                  | 455,250          | 460,941          | 466,780          | 472,536          | 478,443          |
| CENTRAL REGION                                                  | 313,500          | 317,419          | 321,336          | 325,404          | 329,471          |
| WESTERN REGION                                                  | 524,750          | 531,309          | 537,951          | 544,675          | 551,484          |
| INTEREST INCOME                                                 | 15,500           | 15,629           | 15,758           | 15,889           | 16,021           |
| OTHER INCOME                                                    | 12,000           | 12,120           | 12,241           | 12,364           | 12,487           |
| <b>TOTAL INCOME</b>                                             | <b>1,321,000</b> | <b>1,337,417</b> | <b>1,354,039</b> | <b>1,370,869</b> | <b>1,387,906</b> |
| <b>DIRECT COSTS</b>                                             |                  |                  |                  |                  |                  |
| COST OF SALES                                                   | 323,375          | 327,417          | 331,510          | 335,654          | 339,849          |
| DIRECT LABOR                                                    | 129,350          | 130,967          | 132,604          | 134,262          | 135,940          |
| INDIRECT LABOR                                                  | 25,870           | 26,193           | 26,521           | 26,852           | 27,186           |
| COMMISSIONS                                                     | 155,220          | 157,160          | 159,125          | 161,114          | 163,138          |
| ADVERTISING                                                     | 64,675           | 65,483           | 66,302           | 67,131           | 67,970           |

LOC: 5.13 LEN: 10 JUST: R0 DIR: DIR WINDOW: 1 1-1-6  
DATA: 13374174900 FORMULA: 1A..T..11  
COMMAND:

Now that the worksheet is completed, you can begin playing "WHAT IF???". By simply changing any number(s) on your worksheet, you may recalculate another approach to the same problem. This method of analysis saves you many hours of manually testing out a number of approaches to your planning problems. Of course, you are free to change any formula, insert or delete columns or rows (formulas are automatically adjusted if desired), change the name of the worksheet and save a number of variations of the analysis.

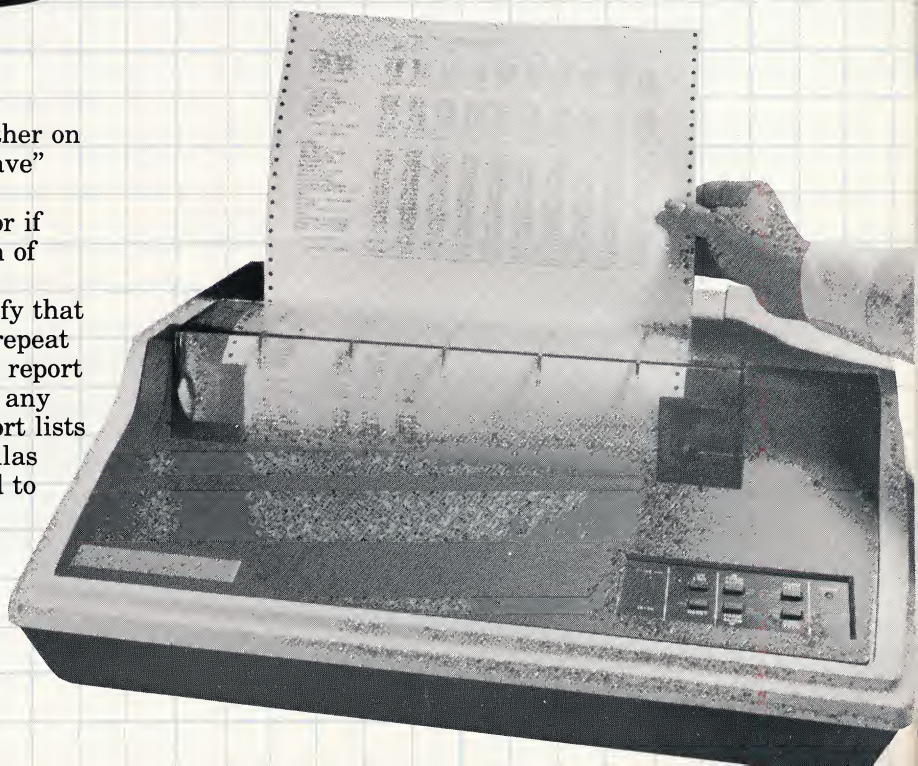
5

| ANNUAL BUDGET FOR AMERICAN PRODUCTS CO.<br>FOR FISCAL YEAR 1983 |                  |                  |                  |  | YEARS<br>TOTAL    |
|-----------------------------------------------------------------|------------------|------------------|------------------|--|-------------------|
|                                                                 | MAR              | APR              | MAY              |  |                   |
| <b>INCOME</b>                                                   |                  |                  |                  |  |                   |
| EASTERN REGION                                                  | 466,780          | 472,536          | 478,443          |  | 5,854,680         |
| CENTRAL REGION                                                  | 321,336          | 325,404          | 329,471          |  | 4,031,723         |
| WESTERN REGION                                                  | 537,951          | 544,675          | 551,484          |  | 6,749,475         |
| INTEREST INCOME                                                 | 15,758           | 15,889           | 16,021           |  | 194,730           |
| OTHER INCOME                                                    | 12,241           | 12,364           | 12,487           |  | 152,190           |
| <b>TOTAL INCOME</b>                                             | <b>1,354,039</b> | <b>1,370,869</b> | <b>1,387,906</b> |  | <b>16,981,798</b> |
| <b>DIRECT COSTS</b>                                             |                  |                  |                  |  |                   |
| COST OF SALES                                                   | 331,510          | 335,654          | 339,849          |  | 4,158,719         |
| DIRECT LABOR                                                    | 132,604          | 134,262          | 135,940          |  | 1,663,488         |
| INDIRECT LABOR                                                  | 26,521           | 26,852           | 27,186           |  | 332,698           |
| COMMISSIONS                                                     | 159,125          | 161,114          | 163,128          |  | 1,996,185         |
| ADVERTISING                                                     | 66,302           | 67,131           | 67,970           |  | 831,744           |

LOC: 5.16 LEN: 10 JUST: R0 DIR: DIR WINDOW: 2 4-1-6  
DATA: 3354037760 FORMULA: 1A..T..9A..125  
COMMAND:

Sometimes, when you have a large worksheet, you may wish to view several sections at the same time. In this example we have defined three "windows" each displaying a different part of the worksheet. You can change one window and watch the results in the others. You may define as many windows as will fit on the screen. Any window may be "locked" to prevent scrolling either vertically or horizontally so you can continue to view it while you scroll through others.

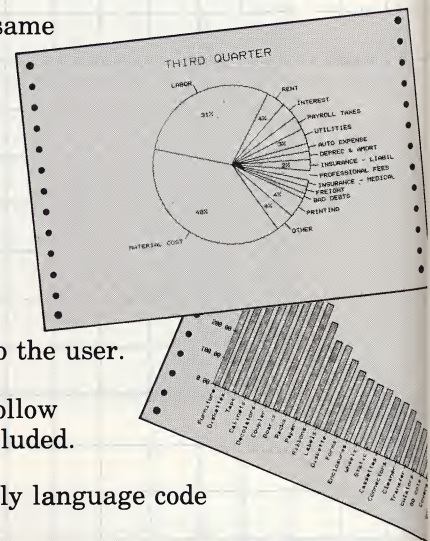
You may choose to print the worksheet either on the line printer or on a terminal driven "slave" printer at any time. Large worksheets will automatically "fold" across multiple pages or if you choose, you may select any combination of columns and rows to print (if the selected worksheet can't fit on a page you may specify that selected heading columns and rows should repeat on multiple pages). There is also an "audit" report available for either the entire worksheet or any combination of columns and rows. This report lists the contents of each location and any formulas entered. It is a valuable tool when you need to audit your worksheet for problems or for documentation justifying your results.





# A Summary of COMPU-SHEET features

- Very easy to use . . . no programming experience is needed.
- Uses simple English-like commands (COPY, MERGE, PRINT . . .), not cryptic command codes.
- In-screen prompting for data . . . the cursor moves in any direction over the worksheet.
- Automatically adjusts formulas during insert and delete functions.
- Formulas provide for "absolute" or "relative" reference to worksheet locations.
- Insert or delete multiple columns or rows with one command.
- User may specify titles to appear anywhere on the worksheet.
- Worksheets may be merged or consolidated. The user controls the range of locations to be merged or consolidated.
- Formulas may specify retrieval of data from other worksheets.
- Formulas may specify retrieval of data from any file in the user's data base (down to the sub-value level, if desired).
- Rapid movement to any point in the worksheet.
- Formulas are validated upon entry. No need to wait until the worksheet is calculated.
- The user may enter a "?" at any point to get a "HELP" display.
- Multiple windows provide for viewing various sections of the worksheet at one time.
- The user controls the width of any column and may change it at any time.
- Any column may be either left or right justified and the masking is set by column (individual locations may override the column mask).
- Worksheets may be password encoded for security.
- The user has the option to control the format of any printed reports. Any combination of columns and rows may be selected.
- Formulas are written in simple algebraic format. Even "IF condition THEN formula1 ELSE formula2" is supported.
- A complete and formatted "AUDIT" report is available.
- All calculations and intermediate results are carried to four decimal place accuracy to insure that rounding errors will not distort the end results (even though displayed values may have less than four decimal places).
- Supports a line printer or terminal driven "slave" printers.
- User coded subroutines may be called from the formulas (for users with technical expertise).
- Multiple versions of the same worksheet may be saved.
- Interfaces with "ACCU-PLOT" graphics system. Your worksheets may be represented as line graphs, pie charts, bar graphs, etc . . .
- Source code is provided to the user.
- A complete and easy-to-follow instruction manual is included.
- Does not use any assembly language code (user modes).
- Operates on Microdata, all PICK systems, and the Prime Information System.
- Operates on any terminal utilizing standard cursor control.
- All revisions and enhancements are provided free of charge for one year.
- A 21 day free trial period is available.
- Available from your local COMPU-SHEET dealer or from: Interactive Systems  
2432 West Peoria Avenue, Suite 1282-B  
Phoenix, Arizona 85029  
(602) 993-3579





# How to order COMPU-SHEET

**You can order COMPU-SHEET with a free 21-day trial period. Here's how:**

- 1) Complete the license agreement below.
- 2) Send the agreement along with your check for \$795.00 to Interactive Systems or your local COMPU-SHEET dealer.
- 3) We will send your copy of COMPU-SHEET (magnetic tape and a complete instruction manual) within 15 days by registered mail.
- 4) We will hold your check for 21 days from the date you receive COMPU-SHEET.

- 5) If for any reason you are not completely satisfied with COMPU-SHEET, within that 21 day period call us and we will authorize you to return COMPU-SHEET. We will then return your UNCASHED CHECK to you.

All revisions and enhancements will be provided to you free of charge for a period of one year from the date of purchase.

**COMPU-SHEET brings the power of the computer to your imagination and fingertips. Complete and mail the license agreement today!**

## COMPU-SHEET NON-EXCLUSIVE SOFTWARE LICENSE AGREEMENT

### Licensee:

Company \_\_\_\_\_

Contact Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ St \_\_\_\_\_ Zip \_\_\_\_\_

Phone \_\_\_\_\_

### Computer Equipment:

Manufacturer \_\_\_\_\_

Model \_\_\_\_\_

Serial number \_\_\_\_\_

Operating system version \_\_\_\_\_

Tape density \_\_\_\_\_

INTERACTIVE SYSTEMS will provide the software product COMPU-SHEET on a non-exclusive perpetual license to the Licensee for use only on the Licensee's Computer System described above. The Licensee understands and agrees that COMPU-SHEET constitutes Proprietary Information and Trade Secrets of INTERACTIVE SYSTEMS and that title and ownership of COMPU-SHEET remains with INTERACTIVE SYSTEMS. The Licensee hereby agrees that COMPU-SHEET may not be copied, sold, rented, leased, given, assigned, or otherwise made available to any other individual or organization not licensed by this agreement. INTERACTIVE SYSTEMS warrants COMPU-SHEET to be free from software logic errors for a period of 1 (one) year. INTERACTIVE SYSTEMS makes no warranty, express or implied, concerning the applicability of this Software for a particular purpose. It is solely the Licensee's responsibility to determine suitability of COMPU-SHEET for a particular purpose. INTERACTIVE SYSTEMS accepts no liability for loss or damage caused, or alleged to be caused, directly or indirectly, by COMPU-SHEET. THE WARRANTIES CONTAINED IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY REGARDING MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

### License Fee

**\$795.00 per system**

Accepted for Licensee by: \_\_\_\_\_

Title: \_\_\_\_\_ Date: \_\_\_\_ / \_\_\_\_ / \_\_\_\_

**INTERACTIVE SYSTEMS ■ 2432 West Peoria Avenue, Suite 1282-B ■ Phoenix, Arizona 85029 ■ (602) 993-3579**

### Ask for information about these other software tools available from INTERACTIVE SYSTEMS:

**TCL-AUDITOR** is a TCL pre-processor which provides each terminal user with a list of the previous 64 sentences entered at TCL. Any of these previous sentences may easily be listed, corrected, modified, expanded, re-executed or saved without re-entering the sentence.

### REPORT-GEN

With REPORT-GEN, you can generate a BASIC program from an ENGLISH sentence, and then modify the program to perform tasks that ENGLISH cannot handle.

### SCREEN-GEN

is a utility that allows *anyone* to develop data entry or inquiry processes such as order entry, journal entry, customer inquiry, etc. SCREEN-GEN includes an ENGLISH-like language that makes *data entry* as easy as ENGLISH makes data retrieval or report generation.



# COMPU-SHEET

**A powerful second-generation  
spreadsheet package from Interactive Systems!**

**For Microdata, all PICK systems, and the Prime  
Information System.**

**Selected by:**

**Bantam Computers for the Bantam 002™ system**

**CDI for each IBM Series/1.**

**Datamedia for the Datamedia 932™ system**

**General Automation for each ZEBRA™ (PICK) system.**

**Available from Interactive Systems and dealers  
around the world.**

For information, contact Interactive Systems  
(602) 993-3579 or your local COMPU-SHEET dealer:



**INTERACTIVE SYSTEMS**

2432 West Peoria Avenue, Suite 1282-B  
Phoenix, Arizona 85029

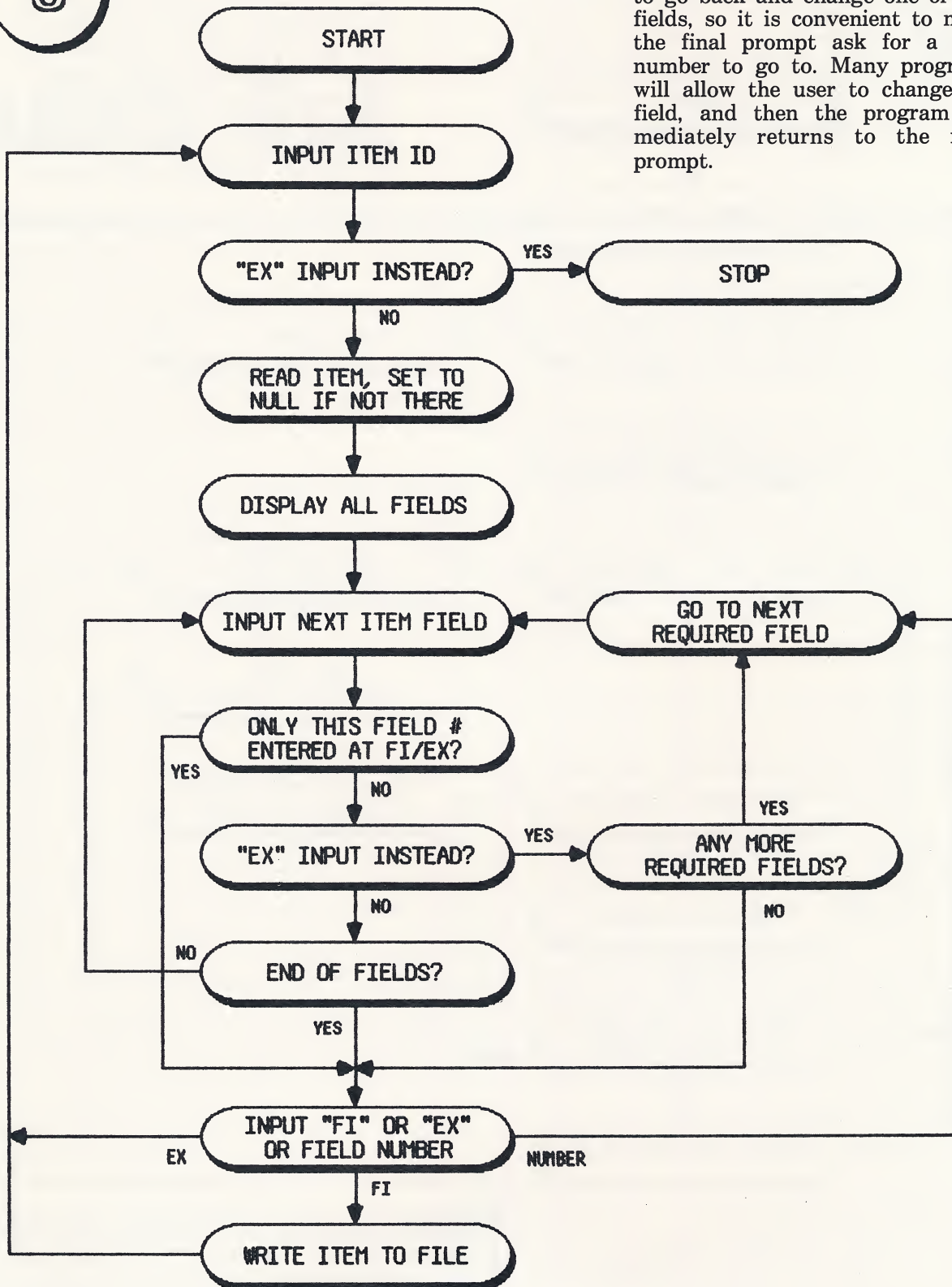
BULK RATE  
U.S. POSTAGE  
PAID  
APTOS, CA 95003  
PERMIT NO. 67

## COMPU-SHEET

**A management tool for company presidents, CPA's, treasurers, controllers, managers,  
or anyone who needs an easier way to solve problems!**



6

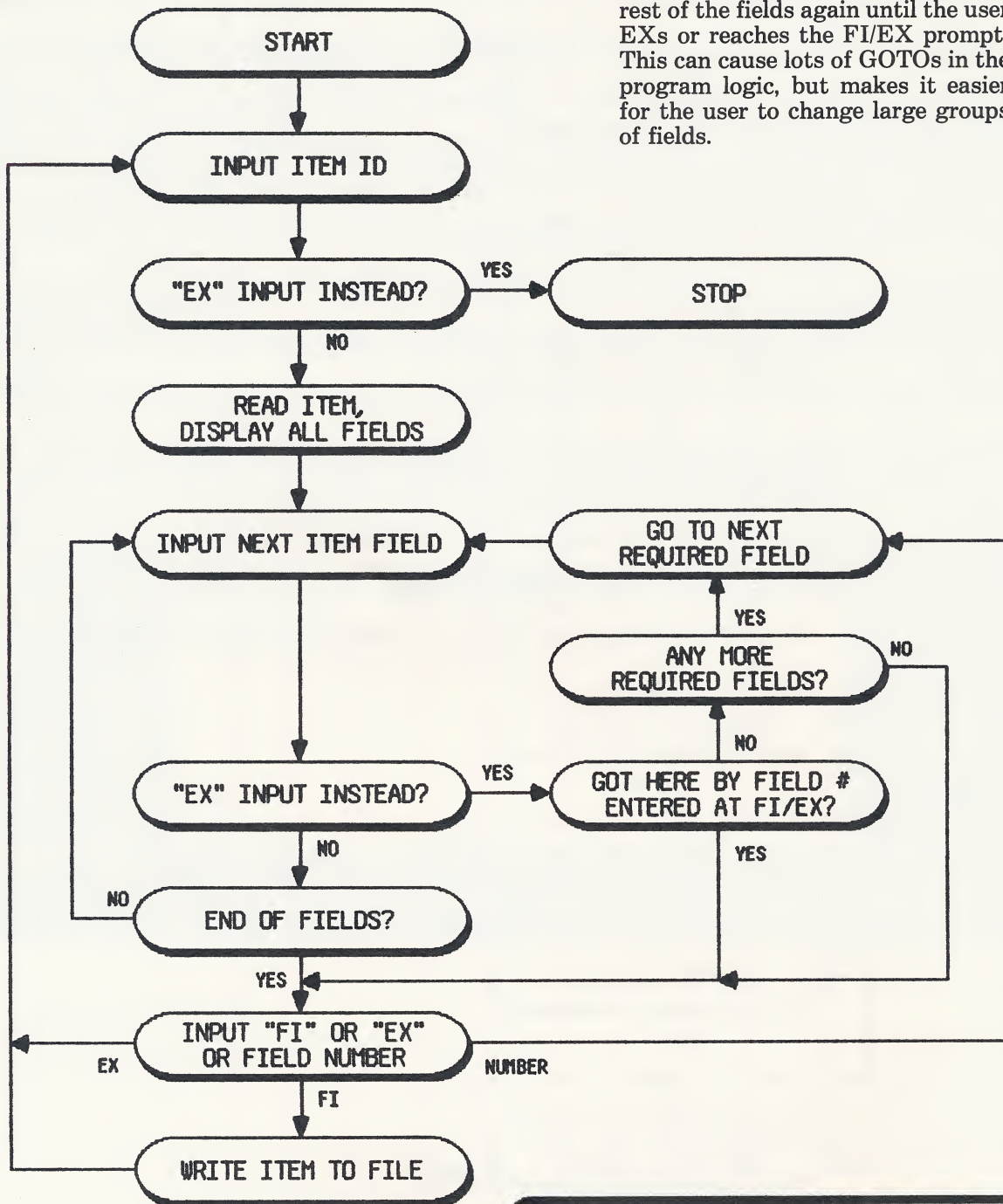


By the time the user gets to the final FI/EX prompt, the user often wants to go back and change one or two fields, so it is convenient to make the final prompt ask for a field number to go to. Many programs will allow the user to change the field, and then the program immediately returns to the final prompt.



7

Instead of immediately returning back to the FI/EX prompt after the user changes a field, the program can just continue on through the rest of the fields again until the user EXs or reaches the FI/EX prompt. This can cause lots of GOTOs in the program logic, but makes it easier for the user to change large groups of fields.



The charts presented have really only addressed the overall flow of control in data entry programs. The formatting and contents of displays, prompts, error messages and commands also greatly affect the style, operation and usefulness of a data entry system. With careful thought and attention to the idea of using a consistent approach, regardless of the level of complexity of each application being addressed, designers and implementors can create clean, easy to use, straight-forward software.

P



# wish list

*The previous 68 Wish List items have been featured in issues #1 through #5 of Pragma.*

**69. Moving Text While Editing.** It is often desirable to move a block of lines from one place to another while editing an item. This can be accomplished with the merge (ME) command, but is somewhat awkward since it must be followed by possibly an F and then a delete (DE). Add a move capability to the editor like the merging capability which currently exists, but which deletes the lines from their original location.

**70. Simplified BREAK Key.** For most users, the BREAK key (when enabled) is always used in conjunction with typing END to abort the current process. Provide a one-keystroke sequence to accomplish the BREAK-END function. Hitting a key twice in a row (or some similar two-keystroke sequence) may be a safer requirement for aborting a process, but any larger number of keystrokes (and being confronted by an asterisk or exclamation point prompt) just tends to become confusing or tedious for users.

**71. Form Feed Before ERRMSG 401.** ERRMSG 401 (NO ITEMS PRESENT) is often lost when output is sent to the printer, because the message is frequently appended to the previous job's listing. Include a form feed in ERRMSG 401 so that it will appear on its own output page.

**72. Writing Transaction Records.** A good application design technique is to have every update program write a history record in a transaction file in order to provide an audit trail. In a multiple-user, multiple-terminal environment, it is necessary to make the record's item identifier be in the PORT-DATE-TIME form to insure uniqueness. Also, some systems will need the file or group to be locked during the write to avoid group format errors, or to allow simultaneous deletions of history records without corrupting the file and creating inconsistent data. To provide those capabilities, a LOCK or READU would have to be executed before creating the history item, but both statements seem clumsy and an overkill for such a simple but frequent operation. Provide a single statement for writing transaction items in locked groups. A PORT-DATE-TIME item identifier would be the default. PORT could include the logged on account name, not just the port number.

**73. CLEAR-FILE Group Locking.** CLEAR-FILE ignores group locks. Make CLEAR-FILE lock each group before clearing it.

**74. TIMESLICE Privileges.** The documentation suggests using the TIMESLICE verb to change timeslices within procs that invoke certain jobs. Unfortunately, the verb requires SYS2 privileges. Allow a timeslice setting capability for accounts without SYS2 privileges.

Users of computer systems should always remember that the nice thing about software (besides the fact that it doesn't break when you drop it) is that programs are relatively easy to change — no soldering gun is necessary. So just because the operating system or the compiler or a system utility happens to work (or fail to work) in some particular fashion now, doesn't mean it has to always be that way. The next time an inspired idea arrives for improving your system, write it down and send it to Pragma for publication in the Wish List.

**75. Blanks in Translate Conversions.** Single valued attributes often have corresponding data stored as a multivalued attribute in another file. The translate (T) conversion code can be used to convert the single value attribute to the corresponding multivalued data, but each multivalued mark is converted to a blank in the process. Provide a translate conversion code that does not change the multivalued mark to a blank.

**76. Command Recall.** Users frequently type long commands which contain a minor syntactic error. If the RETURN key is hit, a command with a minor error must be completely retyped. If no characters have been typed since the last RETURN, make the Control-R key recall the previous command entered, so that the backspace key can be used to fix commands entered in error without having to completely retype the command. Note that this does not conflict with the way in which Control-R already works while a command is being entered, but before RETURN is hit. (This type of capability has already been offered on some system implementations.)

**77. Listing Locks.** The system will allow a program to detect if an execution lock is already set, but there is no way to tell which locks are locked by which ports. Provide a command similar to LIST-LOCKS that shows which execution locks are set by which ports. (This type of capability has already been offered on some system implementations.)

**78. Ignore Null Conversions.** In the report generator, null conversions do nothing, but in BASIC, null conversions generate an error message. Make null conversions in BASIC do nothing (in other words, return the string passed to the conversion routine).

**79. Local BASIC Functions.** The overhead of subroutines (separate compilation, required cataloging, the slow speed of CALLs) is so great that it is often avoided, resulting in messy and obscure programs when just a simple function call capability was all that was needed. Allow user defined functions, with the function body being any number of lines, all of which are contained within one compilation. In BASIC compilers and interpreters outside the Pick marketplace, this is usually accomplished via a DEF statement.

**80. EQUATEs and DIMENSIONs.** A standard programming technique is to define symbolic constants with EQUATE statements for use as limiting values when working with arrays. Unfortunately, EQUATE symbols cannot be used in DIM statements, so that if the EQUATE value is changed, the DIM must be changed, too. Allow EQUATE symbols to be used when defining array dimensions. (This type of capability has already been offered on some system implementations.)

□



# GET, Part 1: An Input Processor

The first in a series of articles discussing input processors is presented. Examples are given showing how code can be reduced by using external tables of input parameters, and the actual format of such a table is described.

At the bottom of the next page are two order entry programs that look completely different, but which do exactly the same thing when executed: prompt for an order number, and then allow the order's description and amount to be created or changed. (See the flowchart on this page.) Why are the two programs so different?

OLD.GET.ORDER is a traditional looking input program for a Pick-style system. It uses lots of PRINT statements for painting the screen, displaying prompts, and showing formatted data after input and conversion. IF statements are used for field validation and plenty of assignment statements help replace old attributes with new data. Lots of apparently arbitrary "magic numbers" are sprinkled throughout the program in order to specify cursor co-ordinates or attribute numbers.

After writing a few dozen such input programs, programmers inevitably begin to notice that all of the programs tend to have many features in common: all input programs need to paint the screen, display prompts, accept input, perform validations and conversions, output formatted data, and so on. Much of the code will appear to be duplicated or repetitive in style, and the programmer will start to create local subroutines for handling the tasks of processing input. For example, a routine might be written that accepts a pair of screen co-ordinates, a prompt string, and a conversion specification as input. The routine would display the prompt at the given co-ordinates, input a response from the operator, perform the given conversion on the input string, and return the result. With such a subroutine, an input program would become primarily a series of GOSUBs, calling the input processing routine for each field on the screen.

Even with such an input subroutine, the programmer is still forced to code many parameters (like co-ordinates and attribute numbers) inside the program. Eventually, the programmer realizes that such constants can all be gathered into a table external to the program, and the subroutine can simply refer to the table in order to perform the necessary input processing.

That is the approach used by the NEW.GET.ORDER version: it uses a table (see the GET.ORDER item alongside the NEW.GET.ORDER program) which is stored in the SCREENS file. Since the table includes all constants such as prompt strings, attribute numbers, conversions and screen co-

ordinates, the program simply calls a subroutine named GET which refers to the table and then handles all input processing. NEW.GET.ORDER needs no PRINT statements and very few assignment or IF statements. Best of all, the magic numbers have all disappeared. In fact, the programmer can simply edit the SCREENS table at any time to redefine prompts or attribute numbers or screen co-ordinates or conversions, and the input program doesn't even need to be recompiled!

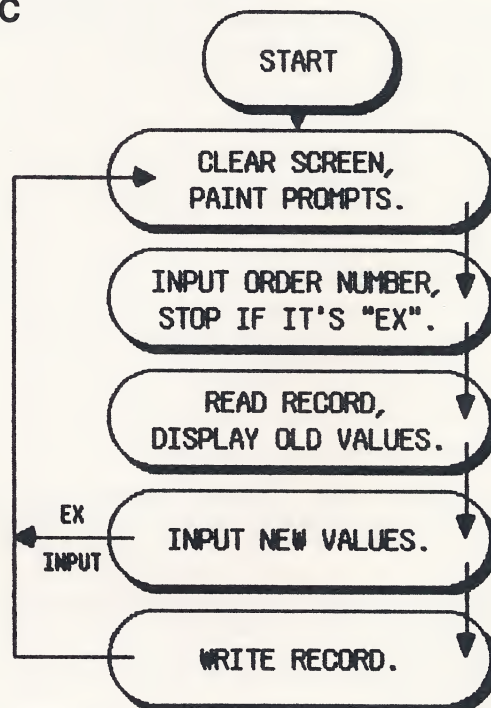
The complete format for tables used by the GET subroutine is documented in the listing on page 30, which describes how each attribute in a table is interpreted. GET can take control from a calling program and automatically process a series of input fields. The parameters for each field are stored as multivalues across attributes in the table. For example, the second field of the GET.ORDER example is the description field, so the second multivalue of each table attribute holds a parameter for that field: the second multivalue of table attribute 16 is the field's output position, the second multivalue of attribute 20 is the output length, the second multivalue of attribute 24 is the input position, and so on. The GET.ORDER table presented here happens to use only a small number of the parameters allowed in a GET table.

An actual CALL to the GET subroutine requires a list of six parameters: (1) the dynamic array in which GET should return all new input, (2) the table holding the field parameters, (3) the dynamic array holding the old data before input, (4) what field number (multivalue) GET should begin with in the table of parameters, (5) a variable in which GET will return the number of the field following the last field processed (so the calling program can know where GET finished in the table), and (6) a variable set by GET to 1 (true) if "EX" was input, otherwise the variable is set to null (this is a convention that allows the calling program to determine if the operator is exiting from the series of input fields).

In the next installment, the actual code for the GET subroutine will be presented.

□

## OLD/NEW.GET.ORDER LOGIC





## TWO EQUAL PROGRAMS

### OLD. GET. ORDER

```

001 OPEN "ORDERS" ELSE STOP
002 PROMPT ""
003 100 *
004 PRINT CHAR(12) : "ORDER NUMBER: " :
005 PRINT @(1, 2) : "DESCRIPTION: " :
006 PRINT @(6, 4) : "AMOUNT: " :
007 PRINT @(14, 0) : ; INPUT ORDER. NUM :
008 IF ORDER. NUM = "EX" THEN STOP
009 ORDER. NUM = ICONV(ORDER. NUM, "MDO")
010 PRINT @(14, 0) : ORDER. NUM "L#10" :
011 READ ORDER FROM ORDER. NUM ELSE ORDER = ""
012 DESC = ORDER<1>
013 AMT = OCONV(ORDER<2>, "MD2")
014 PRINT @(14, 2) : DESC :
015 PRINT @(14, 4) : AMT :
016 PRINT @(14, 2) : ; INPUT NEW. DESC :
017 IF NEW. DESC = "EX" THEN GO TO 100
018 IF NEW. DESC = "" THEN NEW. DESC = DESC
019 PRINT @(14, 2) : NEW. DESC "L#50" :
020 ORDER<1> = NEW. DESC
021 PRINT @(14, 4) : ; INPUT NEW. AMT :
022 IF NEW. AMT = "EX" THEN GO TO 100
023 IF NEW. AMT = "" THEN NEW. AMT = AMT
024 PRINT @(14, 4) : NEW. AMT "L#10" :
025 ORDER<2> = ICONV(NEW. AMT, "MD2")
026 WRITE ORDER ON ORDER. NUM
027 GO TO 100
028 END

```

### NEW. GET. ORDER

```

001 OPEN "ORDERS" TO O. FILE ELSE STOP
002 OPEN "SCREENS" TO S. FILE ELSE STOP
003 READ FIELDS FROM S. FILE, "GET. ORDER" ELSE STOP
004 LOOP
005 CALL GET(ORDER. NUM, FIELDS, "", 1, NEXT. STEP, EXIT)
006 UNTIL EXIT DO
007 ORDER. NUM = ORDER. NUM<1>
008 READ ORDER FROM O. FILE, ORDER. NUM ELSE ORDER = ""
009 CALL GET(ORDER, FIELDS, ORDER, NEXT. STEP, IGNORE, EXIT)
010 IF NOT(EXIT) THEN WRITE ORDER ON O. FILE, ORDER. NUM
011 REPEAT
012 STOP
013 END

```

### GET. ORDER

```

001
002
003 ORDNUMJDESCJAMT
004 Y
005 ORDER NUMBER: \ \ DESCRIPTION:
    \ \ AMOUNT:
006
007 JAMT
008
009
010
011 LJLJL
012 JJJJ2
013 Y
014
015
016 O, 14J2, 14J4, 14
017 10J50J10
018
019 JJMD2
020 10J50J10
021
022
023
024 O, 14J2, 14J4, 14
025
026
027
028
029
030
031 MDOJJMD2
032 FJJF

```

### NEW.GET.ORDER PARAMETERS



# GET SCREEN DEFINITIONS

| File.. | Attribute.. | AMC  | * Description.....                                                                                                                                                                                                     |
|--------|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GF     | SCREEN      | 0    | Screen name, usually the same name as the program that uses the screen. Each screen consists of any number of fields. The screen parameters for each field are stored as multi-values in each of the attributes below. |
| GF     | LABEL       | 3 V  | The label (name) for each field. If there are duplicate labels, only the first will be found by a GOTO. A null label will be treated as the end of the list of field labels.                                           |
| GF     | CLEAR       | 4 V  | "Y" if the screen should be cleared before this field is input.                                                                                                                                                        |
| GF     | TEXT        | 5 S  | Text to display before this field is input. Each subvalue is a row on the screen. Only the first 24 rows are displayed.                                                                                                |
| GF     | LABELS      | 7 S  | The labels of other fields that should be displayed before this field is input.                                                                                                                                        |
| GF     | TYPE        | 9 V  | "N" if no input processing should be done (useful if field is for display only).                                                                                                                                       |
| GF     | JUST        | 11 V | Justification to be used on output. Must be L for left or R for right.                                                                                                                                                 |
| GF     | AMC         | 12 V | Attribute mark count (attribute number) where the data is to be stored in the dynamic array being processed.                                                                                                           |
| GF     | REQ         | 13 V | "Y" if input is required.                                                                                                                                                                                              |
| GF     | OPOS        | 16 V | ROW, COLUMN where the field's output should be placed on the screen.                                                                                                                                                   |
| GF     | OFLN        | 17 V | Output fill length (number of fill characters that should be output before the data, to clear the field).                                                                                                              |
| GF     | OFILL       | 18 V | Output fill character. If null, a blank is used.                                                                                                                                                                       |
| GF     | OCONV       | 19 S | List of conversions to perform on data before it is output.                                                                                                                                                            |
| GF     | OLEN        | 20 V | Output length (number of characters to limit the output to).                                                                                                                                                           |
| GF     | COPY        | 21 V | If this parameter is null, the user is prompted for data. Otherwise, this indicates what other attribute to copy to create the data, and the user is not prompted at all.                                              |
| GF     | PROMPT      | 22 V | Prompt text to display before this field is input.                                                                                                                                                                     |
| GF     | PPOS        | 23 V | ROW, COLUMN prompt position.                                                                                                                                                                                           |
| GF     | IPOS        | 24 V | ROW, COLUMN input position.                                                                                                                                                                                            |
| GF     | IFLEN       | 25 V | Input fill length.                                                                                                                                                                                                     |
| GF     | IFILL       | 26 V | Input fill character. If null, a blank is used.                                                                                                                                                                        |
| GF     | ICONV       | 31 S | List of conversions to perform on data after it is input.                                                                                                                                                              |
| GF     | FILE        | 32 V | "F" if, after this field is input, screen processing should stop and return all data captured so far. The last field must have an "F" here.                                                                            |
| GF     | GOTO        | 37 V | Label of field to be input next. If null, the immediately following field is input next. (P)                                                                                                                           |



# local user groups

Where are the local user groups and what are they up to? This regular department is designed to help our readers stay aware and informed of nearby groups and activities for computer users.

Why isn't YOUR user group mentioned below? To keep us informed of your group's activities and to appear in this department in future issues, keep *Pragma* on your mailing list and send us your group's newsletter!

## Northern California

The Northern California Pick Users is perhaps the fastest growing user group that reports to *Pragma*, since their six page monthly newsletter almost always reports a good number of new members that have joined this organization. The attendance record for the group's monthly meetings was broken in September, when over 110 users showed up to hear speaker Bill Walsh of Pick Systems. A transcript of that meeting is available for \$10 from Secretary Lisa Levsen at Box 6759, Oakland, CA 94603. The presentation at the group's October meeting was on "effective time utilization", while the legal implications of computer contracts will be the presentation at the November 15th meeting in Oakland. Call Lisa at 415-632-0977 for more information.

## Delaware Valley

Our most recent letter from the Delaware Valley Data Base Management Association included a list of over forty companies that form this group. The post-summer season kickoff meeting was held in September and featured panel discussions on contract negotiation and hardware repair services (moderated by Paul Billiard), while the October agenda included discussions on benchmarks and use of the spooler. Instructional seminars are scheduled for the upcoming November meeting. Users interested in joining this group should contact President Jim Cates, in care of Pars Manufacturing, Box 149, Amber, PA 19002.

## Arizona

The schedule of educational sessions given by members of this group has included a discussion on the similarities and differences between Pick and Microdata systems, led by Jim Tyron in August, and a September session on communications by Bill Bertovich. A survey of members has revealed that system and programming efficiency, followed by system utilities and product updates, are the top three topics users want to learn more about at future meetings. The group currently distributes meeting minutes each month, but is considering publishing a regular newsletter. Users interested in more information about the Pick Users Group of Arizona should contact Secretary Jodi Hilgenberg, Communication Skill Builders, Box 42050, Tucson, AZ 85733.

## Washington DC

Members of the National Capital Area Microdata Users Group finished their third consecutive monthly meeting devoted to discussions of the Sequel in July, enjoyed a hands-on demonstration of the Zebra system (arranged by Bill Cook of General Automation) in September, and were given a presentation on third party hardware maintenance in October. Dues

are \$25 for active members, \$50 for associate members. Contact Stan Seidlitz for more information by writing Continental Computer Services, 9489 Silver King Court, Fairfax, VA 22031 or telephoning 703-273-0816.

## Southern California

The California Data Base Management Association (CDBMA) chose Al Thompson as Member of the Year at CDBMA's August meeting in Newport Beach, which featured a presentation on the Revelation operating system by Roger Harpel of Cosmos. The September meeting in Carlsbad included speaker Ken Trater of SMI, discussing that company's IBM-based Pick products, while the October meeting began with a technical session by John Timmons on the "hidden features of the Pick system" and finished with a speaker on voice response, encoding and recognition. The next meeting is scheduled for November 16th in Irvine and will offer a technical session on A-correlatives. A one day seminar including hardware demonstrations is being planned for January. Dues for this group are \$25, and monthly dinner meeting reservations are \$18 for members, \$23 for non-members. Linda Ristow is the Secretary of this group, which can be reached by writing CDBMA, 9740 Appaloosa Road, San Diego, CA 92131 or by calling 619-578-3152. P

## Some New Subscribers

**Jack L. Runyan**  
Ewell Industries Inc.  
Lakeland, FL

**Barry M. Harriss**  
North Florida Data Service  
Palatka, FL

**Kenneth Scott Akom**  
James McGraw Inc.  
Richmond, VA

**R. Paul Crafts**  
Valley Industries Inc.  
St. Louis, MO



# command files

Command files are typically the "glue" that holds the components of a software system together. Microdata's command file capability is called PROC, Prime offers Perform, and so on. In this regular department, Pragma will be presenting concepts and techniques to help installations squeeze the most out of their command file processors.

## Do You Know Your Proc Limits?

A short quiz, designed to separate the novices from the gurus, is presented for proc programmers.

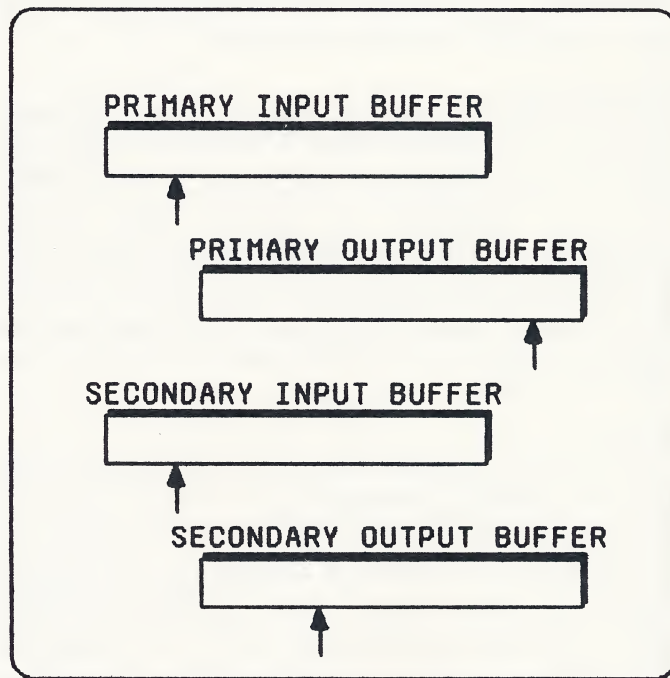
As Mike Schellenbach reminds us at the end of this issue's *Producer Profile* on page 18, "a file can have any number of records, and any record can be virtually any size and have any number of fields and have any number of subfields in those fields, and the length of any of those things is irrelevant". Wouldn't it be nice if every aspect of the operating system was designed in such a general fashion? Unfortunately, programmers often run into seemingly arbitrary software limitations while working with various portions of the operating system, and proc is no exception.

Some restrictions, such as the maximum item size, are well known. Others are obscure and rarely encountered. What follows is a short quiz, compiled from the Microdata™ *Proc Programming Manual*, to test your knowledge of some of the limitations imposed by proc. If you can answer every question correctly, consider yourself a true proc guru.

1. How many blanks may immediately follow a numeric label on a proc statement?
2. What is the longest that a proc can be?
3. How many select registers are there?
4. How many characters may be moved to the secondary output buffer before a "continuation sequence" must be included?
5. How many group locks can be set with F-UREAD commands?

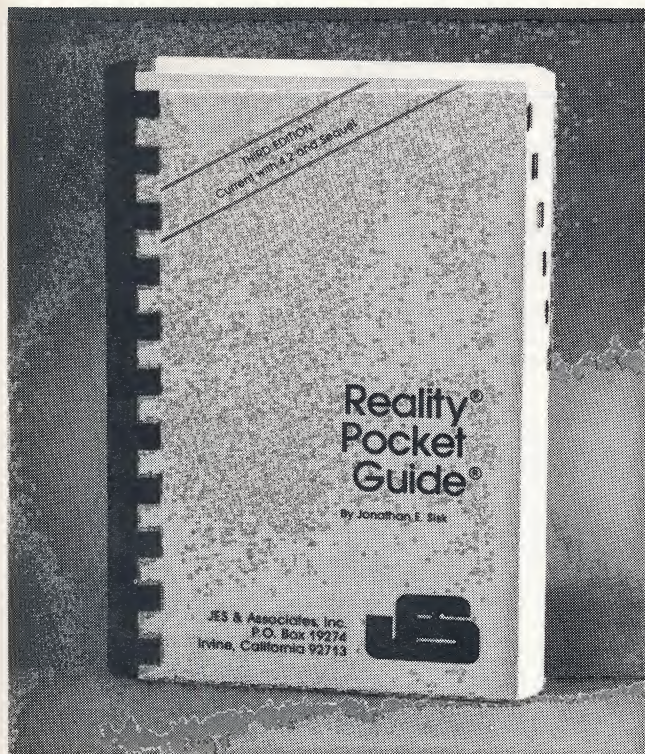
6. What is the maximum size tape record that can be read by the proc IT command?
7. How many values can be stacked with the F; command?
8. How many execution locks can be set with P commands?
9. How many file buffers are there?
10. What is wrong with the following illustration of pointers to the four buffers, as they were supposedly positioned during the execution of a proc?

P



|                      |                                                            |
|----------------------|------------------------------------------------------------|
| ANSWERS              | 6. 800                                                     |
| 1. 1                 | 7. 28                                                      |
| 2. 32,267 characters | 8. 64                                                      |
| 3. 5                 | 9. 9                                                       |
| 4. 140               | 10. Only one input buffer pointer at a time is maintained. |
| 5. 32                |                                                            |





THE NEW SOFT COVER EDITION

## OUR PRODUCTS MAKE YOUR COMPUTER EASIER TO USE.

From Jonathan E. Sisk, author of the original "REALITY POCKET GUIDE" and the "PICK POCKET GUIDE" comes the next generation of quick-reference products that describe every command pertaining to the following topics:

- EDITOR
- DATA/BASIC
- ENGLISH
- PROC
- RUNOFF
- WORDMATE
- TCL
- SYSTEM DEBUGGER
- DATA/BASIC DEBUGGER
- SPOOLER
- ERRMSG
- ASCII

- **THE REALITY POCKET GUIDE, Hard-Cover Edition** ..... **\$29.95**  
The original, now in its third printing, is current with Release 4.2. The 6-ring binder fits easily into a briefcase and lays flat on any surface for easy use. Colored index tabs make access to any section a snap.
- **THE REALITY POCKET GUIDE, Soft-Cover Edition** ..... **\$19.95**  
(SHOWN ABOVE). The same contents as the hard-cover edition, but packaged in a spiral binder.
- **THE REALITY HELP PROCESSOR** ..... **\$95.00**  
The on-line version of the REALITY POCKET GUIDE provides instant access to over 600 help screens by simply entering the word, HELP, followed by the desired command. Each help screen displays the format of a command and a brief explanation of its use. Provided on all types of media.
- **THE PICK HELP PROCESSOR** ..... **\$95.00**  
The on-line version of the PICK POCKET GUIDE, for users of any system running the PICK Operating System. Same functions as the REALITY HELP PROCESSOR. Provided on all types of media.

| Product                                  | Unit Price | QTY.    | Extended Price |
|------------------------------------------|------------|---------|----------------|
| Reality Pocket Guide, Hard-Cover Edition | \$29.95    | x _____ | = _____        |
| Reality Pocket Guide, Soft-Cover Edition | 19.95      | x _____ | = _____        |
| Reality Pocket Guide, Edition 3 Update   | 10.00      | x _____ | = _____        |
| Reality Help Processor                   | 95.00      | x _____ | = _____        |
| Pick Help Processor                      | 95.00      | x _____ | = _____        |

#### Shipping, Per Order

|                              |         |       |       |
|------------------------------|---------|-------|-------|
| Books, Within United States  | \$ 2.50 | ..... | _____ |
| Books, Outside United States | 7.50    | ..... | _____ |
| Tapes, Within United States  | 10.00   | ..... | _____ |
| Tapes, Outside United States | 25.00   | ..... | _____ |

California State Tax (6%) \_\_\_\_\_

**Total Enclosed** \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Phone \_\_\_\_\_ Contact \_\_\_\_\_

Computer Manufacturer \_\_\_\_\_ 800/1600 BPI \_\_\_\_\_

☐ Mastercharge ☐ VISA

☐ Charge my credit card account for \$ \_\_\_\_\_ Expiration Date: \_\_\_\_/\_\_\_\_

Credit Card No. | | | | | | | | | | | | | | | | | | | | | |

Signature: \_\_\_\_\_

Please make checks payable to:

**JES & Associates, Inc.**  
P.O. Box 19274  
Irvine, CA 92714  
(714) 786-2211





# VANILLA (The No-Frills Manufacturing System) Part 6: Inspection

**This sixth installment in a series on the design and implementation of a manufacturing system presents the program for inspecting and then accepting or rejecting purchase order receipts.**

The previous article in this series [*Pragma* #5, page 20] presented the RECEIVE program for entering purchase order

receipts. The INSPECT program listed here on pages 36 and 37 is the next step in purchase order processing, in that INSPECT is designed to record the inspection of purchase order receipts, by either accepting parts and increasing on hand inventory counts, or by rejecting parts for future disposition by a material review board (MRB).

The file formats shown below include descriptions of IM file and PO file attributes for tracking inspection and MRB quantities, both of which are updated by the INSPECT program. (See previous installments for complete descriptions of the other IM and PO file attributes.) Also listed is the complete format of the new MRB file that INSPECT maintains.

The chart on the opposite page documents the interaction of all Vanilla programs and files presented to date. The incomplete handling of the BM and VM files is given away by the lack of arrows around those files on the diagram. Once work order processing is supported, the bill of material file will become important for the product structure information stored there. The vendor file requires a GET.VM edit program, which will be provided in a future installment.

The next Vanilla installment will address the disposition of parts rejected to MRB.

## FILE FORMATS

```
File.. Attribute.. AMC * Description.....
```

```
IM                               Inventory master.
IM      MRB. QTY                 3 A Quantity in MRB.
```

\*\*\*

|     |        |     |                              |
|-----|--------|-----|------------------------------|
| MRB |        |     | Material Review Board parts. |
| MRB | TICKET | 0   | Ticket number.               |
| MRB | DATE   | 1 A | Date of rejection.           |
| MRB | PO     | 2 A | Purchase order number.       |
| MRB | ITEM   | 3 A | Purchase order item number.  |
| MRB | QTY    | 4 A | Quantity rejected.           |

\*\*\*

|    |           |      |                                                 |
|----|-----------|------|-------------------------------------------------|
| PO |           |      | Purchase orders.                                |
| PO | INSP. QTY | 23 V | Quantity inspected.                             |
| PO | TO. MRB   | 24 V | Quantity sent to MRB from receiving inspection. |

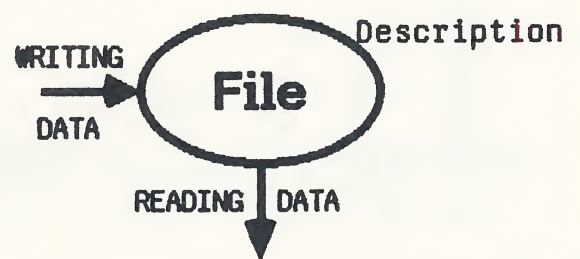
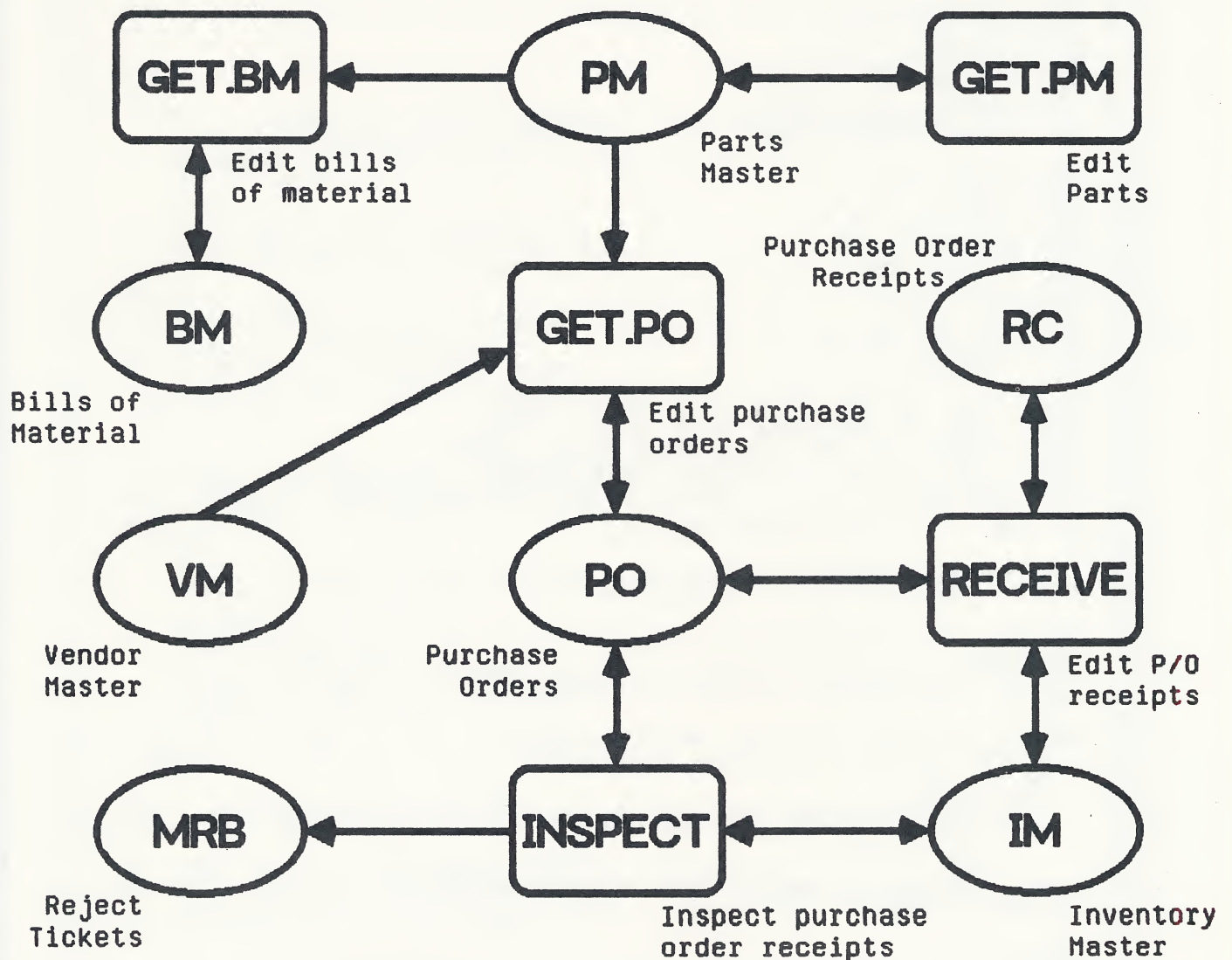
## SCREEN DEFINITION

[illegible]



# VANILLA Files and Programs

## (to date)





# INSPECT PROGRAM LISTING

```

001 EQU IM$ON. HAND TO 1
002 EQU IM$RI. QTY TO 2
003 EQU IM$MRB. QTY TO 3
004 *
005 EQU MRB$DATE TO 1
006 EQU MRB$PO TO 2
007 EQU MRB$ITEM TO 3
008 EQU MRB$QTY TO 4
009 *
010 EQU PO$VENDOR TO 1
011 EQU PO$QA TO 5
012 EQU PO$ITEM TO 7
013 EQU PO$PART TO 8
014 EQU PO$DESC TO 15
015 EQU PO$ACCOUNT TO 17
016 EQU PO$QTY. IN TO 20
017 EQU PO$INSP. QTY TO 23
018 EQU PO$TO. MRB TO 24
019 *
020 EQU item TO 2
021 EQU remain TO 3
022 EQU checked TO 4
023 EQU accepted TO 5
024 EQU mrbqty TO 6
025 EQU ticket TO 7
026 EQU part TO 8
027 EQU desc TO 9
028 EQU vendor TO 10
029 EQU command TO 11
030 *
031 OPEN "DF" TO DF. FILE ELSE STOP "VAN1"
032 OPEN "IM" TO IM. FILE ELSE STOP "VAN2"
033 OPEN "MRB" TO MRB. FILE ELSE STOP "VAN3"
034 OPEN "PO" TO PO. FILE ELSE STOP "VAN4"
035 READ SCREEN FROM DF. FILE, "#INSPECT" ELSE STOP "VAN5"
036 *
037 AT. ERR = @(0,23):CHAR(27):"K"
038 *
039 100 *
040 PRINT CHAR(12):@(59):TIMEDATE():
041 *
042 200 *
043 RELEASE
044 INPUT PO. ID USING SCREEN, "" ELSE STOP
045 PO. ID = PO. ID<1>
046 *
047 READU PO FROM PO. FILE, PO. ID LOCKED
048 PRINT AT. ERR: "PO group locked!":
049 GO TO 200
050 END ELSE
051 PRINT AT. ERR: "PO not found!":
052 GO TO 200
053 END
054 INPUT IGNORE USING SCREEN, PO<PO$VENDOR> AT vendor ELSE NULL
055 PRINT AT. ERR:
056 *
057 300 *
058 INPUT ITEM USING SCREEN, "" AT item ELSE GO TO 100
059 ITEM = ITEM<1>
060 IF PO<PO$ITEM, ITEM> # ITEM THEN
061 PRINT AT. ERR: "Item not found!":
062 GO TO 300
063 END
064 *
065 QTY. IN = 0
066 I = 1
067 LOOP UNTIL PO<PO$QTY. IN, ITEM, I> = "" DO
068 QTY. IN = QTY. IN + PO<PO$QTY. IN, ITEM, I>
069 I = I+1
070 REPEAT
071 REMAIN = QTY. IN - PO<PO$INSP. QTY, ITEM>
072 PART = PO<PO$PART, ITEM>
073 QA. PART = ((PART # "") AND (PO<PO$QA>="Y"))

```



```

074 INV.PART = ((PART#"" ) AND (PO<PO$ACCOUNT,ITEM> = "1234-56"))
075 IF (PART = "" ) OR (NOT(GA.PART)) OR (REMAIN <= 0) THEN
076     PRINT AT.ERR:"Item has nothing to inspect!":
077     GO TO 300
078     END
079 INPUT IGNORE USING SCREEN, PART AT part ELSE NULL
080 INPUT IGNORE USING SCREEN, PO<PO$DESC,ITEM> AT desc ELSE NULL
081 READU IM FROM IM.FILE, PART LOCKED
082     PRINT AT.ERR:"Part group locked!":
083     GO TO 300
084     END ELSE IM = ""
085 INPUT IGNORE USING SCREEN, REMAIN AT remain ELSE NULL
086 PRINT AT.ERR:
087 *
088 400 *
089 LOOP
090     INPUT INSPECTED USING SCREEN, "" AT checked ELSE
091     INPUT IGNORE USING SCREEN, "" AT remain ELSE NULL
092     RELEASE IM.FILE, PART
093     GO TO 300
094     END
095     INSPECTED = INSPECTED<1>
096 WHILE (INSPECTED > REMAIN) DO
097     PRINT AT.ERR:"There isn't that much to inspect!":
098 REPEAT
099 PRINT AT.ERR:
100 INPUT ACCEPTED USING SCREEN, INSPECTED AT accepted ELSE GO TO 400
101 ACCEPTED = ACCEPTED<1>
102 IF ACCEPTED > INSPECTED THEN
103     PRINT AT.ERR:"There isn't that much to accept!":
104     GO TO 400
105     END
106 *
107 REJECTED = INSPECTED - ACCEPTED
108 INPUT IGNORE USING SCREEN, REJECTED AT mrbqty ELSE NULL
109 TICKET = ""
110 IF REJECTED > 0 THEN
111     LOOP
112         INPUT TICKET USING SCREEN, "" AT ticket ELSE GO TO 400
113         TICKET = TICKET<1>
114         UNUSED = 0
115         READU MRB FROM MRB.FILE, TICKET ELSE UNUSED = 1
116         UNTIL UNUSED DO
117             PRINT AT.ERR:"Ticket already exists!":
118             RELEASE MRB.FILE, TICKET
119         REPEAT
120         PRINT AT.ERR:
121         END
122     *
123 INPUT IGNORE USING SCREEN, "" AT command ELSE GO TO 100
124 BREAK KEY OFF
125 *
126 IM<IM$RI.QTY> = IM<IM$RI.QTY> - INSPECTED
127 IF INV.PART THEN IM<IM$ON.HAND> = IM<IM$ON.HAND> + ACCEPTED
128 IM<IM$MRB.QTY> = IM<IM$MRB.QTY> + REJECTED
129 WRITE IM ON IM.FILE, PART
130 *
131 PO<PO$INSP.QTY,ITEM> = PO<PO$INSP.QTY,ITEM> + INSPECTED
132 PO<PO$TO.MRB,ITEM> = PO<PO$TO.MRB,ITEM> + REJECTED
133 WRITE PO ON PO.FILE, PO.ID
134 *
135 IF REJECTED > 0 THEN
136     MRB = ""
137     MRB<MRB$DATE> = DATE()
138     MRB<MRB$PO> = PO.ID
139     MRB<MRB$ITEM> = ITEM
140     MRB<MRB$QTY> = REJECTED
141     WRITE MRB ON MRB.FILE, TICKET
142     END
143 *
144 BREAK KEY ON
145 GO TO 100
146 *
147 END

```

**INSPECT** LISTING CONTINUED



**How Can  
Your Direct  
Mail Flyer  
Reach  
Thousands  
of Pick  
Users for  
Only 34¢  
Per Name?**

**To find out,  
call Pragma today!  
408-688-9200**

**LET MADIC HELP YOU CONTROL  
YOUR MANUFACTURING BUSINESS**

ON-LINE  
INTEGRATED  
ADAPTABLE

CORPORATE  
PLANNING

INVENTORY  
CONTROL

BILLS OF  
MATERIAL

CONFIGURATION  
CONTROL

WORK IN  
PROCESS

QUALITY  
ASSURANCE

PURCHASING

MARKETING

ACCOUNTS  
PAYABLE

ACCOUNTS  
RECEIVABLE

FIELD  
SERVICE

MATERIAL  
REQUIREMENTS

CAPACITY  
PLANNING

COST  
ACCOUNTING

PROJECT  
CONTROL

PLANT  
MAINTENANCE

GENERAL  
LEDGER

PAYROLL

PERSONNEL

**MADIC** 3960 Freedom Circle Santa Clara, CA 95054 (408) 988-8311

Call Today  
Or Send Us Your  
Business Card  
For More Information

# **DOLARS™**

**Distributed On-Line Accounting and Reporting System**

## **M A K E S S E N S E**

*The BEST Financial Software for the PICK Operating System*

### **DELIBERATE SYSTEMS**

**3456 Camino Del Rio North • San Diego, CA 92108  
(619) 280-4400**



# queries

*The following Query was originally published with no available answer in Pragma #2, page 31.*

**12. When the WHERE (S) command is given, a column showing LINKS is output, but the documentation does not explain what LINKS are. What are LINKS?**

LINKS are four bytes from the Process Identification Block (PIB) for each port. (See the chart describing the PIB in the Assembly Language Programming Manual.) The first byte of LINKS (byte two of the PIB when counting from zero) is the byte address of the last character in the terminal input/output buffer. The second byte is the number of bytes in the buffer minus one. The next byte is the link to the previous PIB, and the last byte is the link to the next PIB. *[Answer submitted by Ken Baker, ADP Network Services, Dallas, TX]*

*Of the 25 Queries that have been featured in issues #1 through #5 of Pragma, six Queries now remain unanswered.*

**26. What is POKE?**

POKE is an undocumented verb that allows one port to input commands for immediate execution at another port. The POKE verb can be defined in the Master Dictionary as an item with a P in attribute one, and 0186 in attribute two. When the POKE verb is executed, it will first prompt for a CHANNEL (port number) and then for a COMMAND. The command you input will then be executed by the system as though it were originally input at the port you specified. For example, if you are at port 5, execute POKE, and then answer 13 in response to the CHANNEL prompt and OFF in response to the COMMAND prompt, port 13 will then logoff as though OFF were typed directly at port 13. Instead of a command, certain control characters can be entered for some special effects: Control-L will cause the COMMAND prompt to repeat to allow multiple commands to be easily entered for one port, Control-E will end the loop started by Control-L, Control-C will return directly to the CHANNEL prompt, Control-D will send the port to the debugger (if allowed), Control-S will disable a port that is not logged on so that an account and password cannot be entered, Control-I will enable a port disabled with Control-S, and Control-Z will unconditionally send a port to the system debugger. Since POKE is undocumented, don't assume all of its features work as explained above on your particular machine, or that all of its features have been discovered. Also note that many terminals intercept and recognize certain control characters (Control-D might disable the keyboard, for example), so not every terminal is capable of exercising all POKE commands.

Ⓟ

## Proven Pick System Software

for

- Order Entry/Distribution
- Property Management/Real Estate
- Staff/Resource Scheduling
- General Accounting
- Site Selection

from

**CTS Services, Inc.**  
**2720 Stemmons Freeway**  
**Suite 805**  
**Dallas, TX 75207**  
**214/637-3344**

## FIXED ASSET/PREPAID EXPENSE ACCOUNTING MODULE

By Diogenes Computer Systems, Inc.  
 Specialists — Properties Management  
 Real Estate Construction

- \* Easy to use
- \* Hardcopy Validation/Backup
- \* Flexible Accounting Periods
- \* Book, Tax and State Calculations
- \* Supports all IRS Approved Methods
- \* 2 Years of Proven Reliability
- \* Built in Security
- \* Complete User and Program Documentation
- \* Stand-Alone or Integrate with your G/L
- \* Includes 1981 Economic Recovery Tax Act
- \* Recapture Calculated Automatically
- \* Depreciation Forecasting Module
- \* Demonstrations Available in New York Metropolitan Area

**AFFORDABLY PRICED AT \$2500.00**

Integration and Implementation Available

For More Information —

Call — (201) 764-7282 8 A.M.—8 P.M.

Write — Diogenes Computer Systems, Inc.  
 P.O. Box 667  
 Highland Lakes, N.J. 07422

Software Licensing Inquiries Invited



# Individual Accounts...

The pros and cons of structuring a file system as shared accounts or as individual accounts are briefly described. Charts diagramming the two methods are presented.

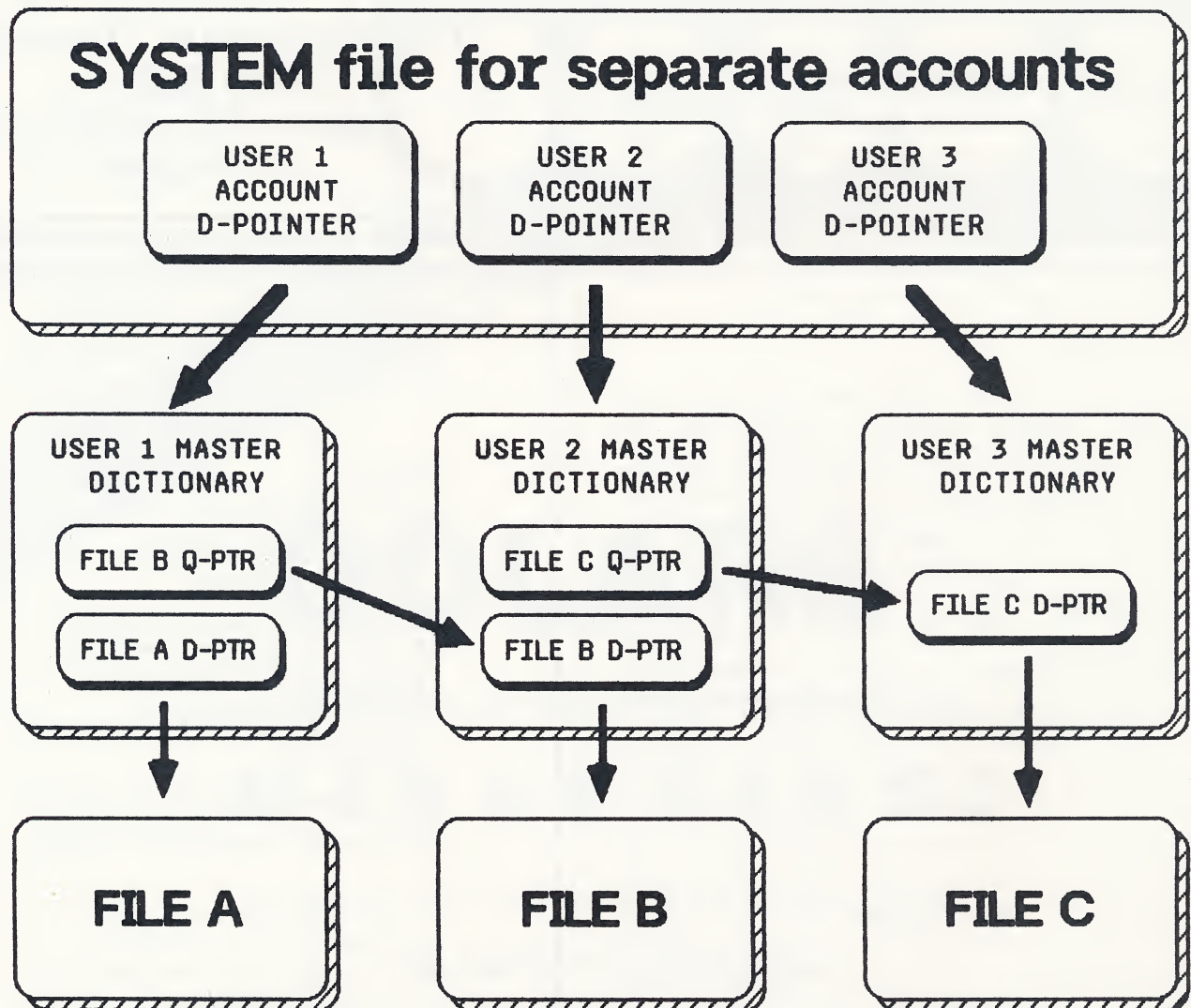
The CREATE-ACCOUNT proc is possibly overused at some installations, since it is such a convenient way to quickly set up a new account. The alternative of defining the new account as a synonym for an old existing account should perhaps be considered instead.

The two diagrams shown below illustrate the two different ways in which accounts can be established. The chart on the left shows the usual method, a result of repeatedly using CREATE-ACCOUNT: each account has a definition item in

the SYSTEM dictionary, and each item is a D-pointer to a separate Master Dictionary for each account. (For an explanation of pointers and how they are used, see *The Ubiquitous POINTER-FILE* in this issue on page 6.) In this scheme, if one account requires access to a file defined in another account, the account needing access must have a Q-pointer in its own Master Dictionary, pointing to the other account's file.

Besides being easy to establish such accounts because of the convenience of CREATE-ACCOUNT, the other main advantage of separate account organization is that it is relatively easy to keep verbs and files out of the reach of unauthorized users on other accounts. Also, STAT-FILE statistics are broken down by account, which may help identify interesting facts regarding system usage, overhead, activity, and so on.

The primary disadvantage of separate accounts is that there is potentially a large amount of data redundancy since each account will have its own copy of all verbs and other Master Dictionary items, and perhaps even separate copies of some files or data items. Multiple copies of the same BASIC program often end up spread over separate accounts, for example.





# Or Shared Accounts?

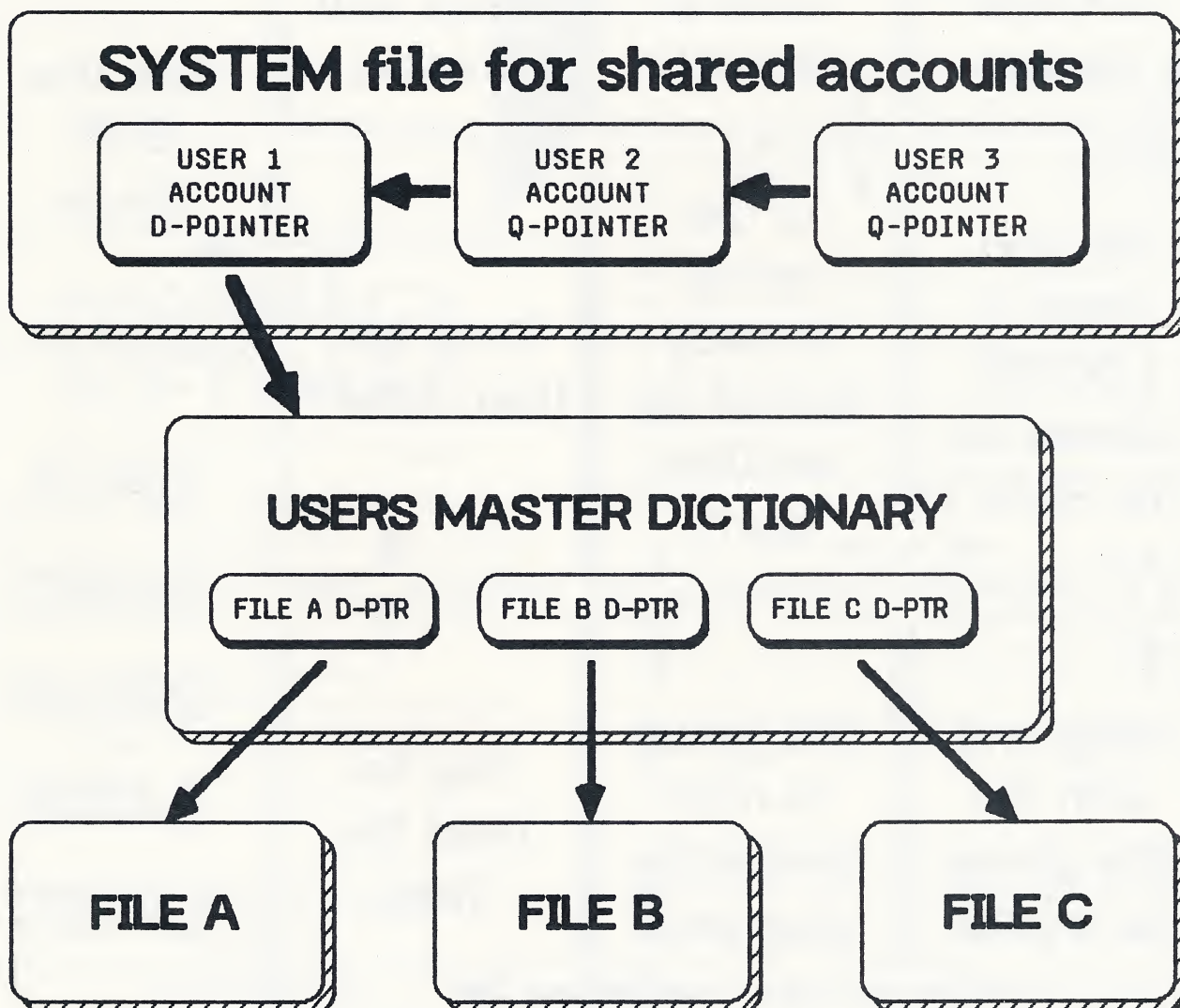
Another disadvantage is that the Master Dictionaries can become quite tangled with a clutter of Q-pointers, trying to reach files defined in other accounts.

The chart on the right shows a different method: each account has a definition item in the SYSTEM dictionary, but two items are Q-pointers to the first item, which is the only actual D-pointer defining a Master Dictionary. In this scheme, all the accounts share one Master Dictionary (even though the SYSTEM items may define different passwords and privileges for each account).

The main advantage of shared account organization is that there is typically very little data redundancy since there is only one Master Dictionary, containing no Q-pointers at all, and there are never any separate copies of files or data items. The result is usually an improvement in system performance and throughput. And as each new Q-pointer account is added, the one Master Dictionary should continue to suffice, avoiding yet more pointers and use of disk space. The ability to essentially backup all accounts in ACCOUNT-SAVE form may also be an advantage.

The primary disadvantage of shared accounts is that more care and effort is required to keep verbs and files out of the reach of unauthorized users. Also note that STAT-FILE statistics are no longer broken down by account under this scheme.

The next time it is necessary to establish a new account at your site, be sure to compare these two diagrams in order to select the method most appropriate for your installation. [P]





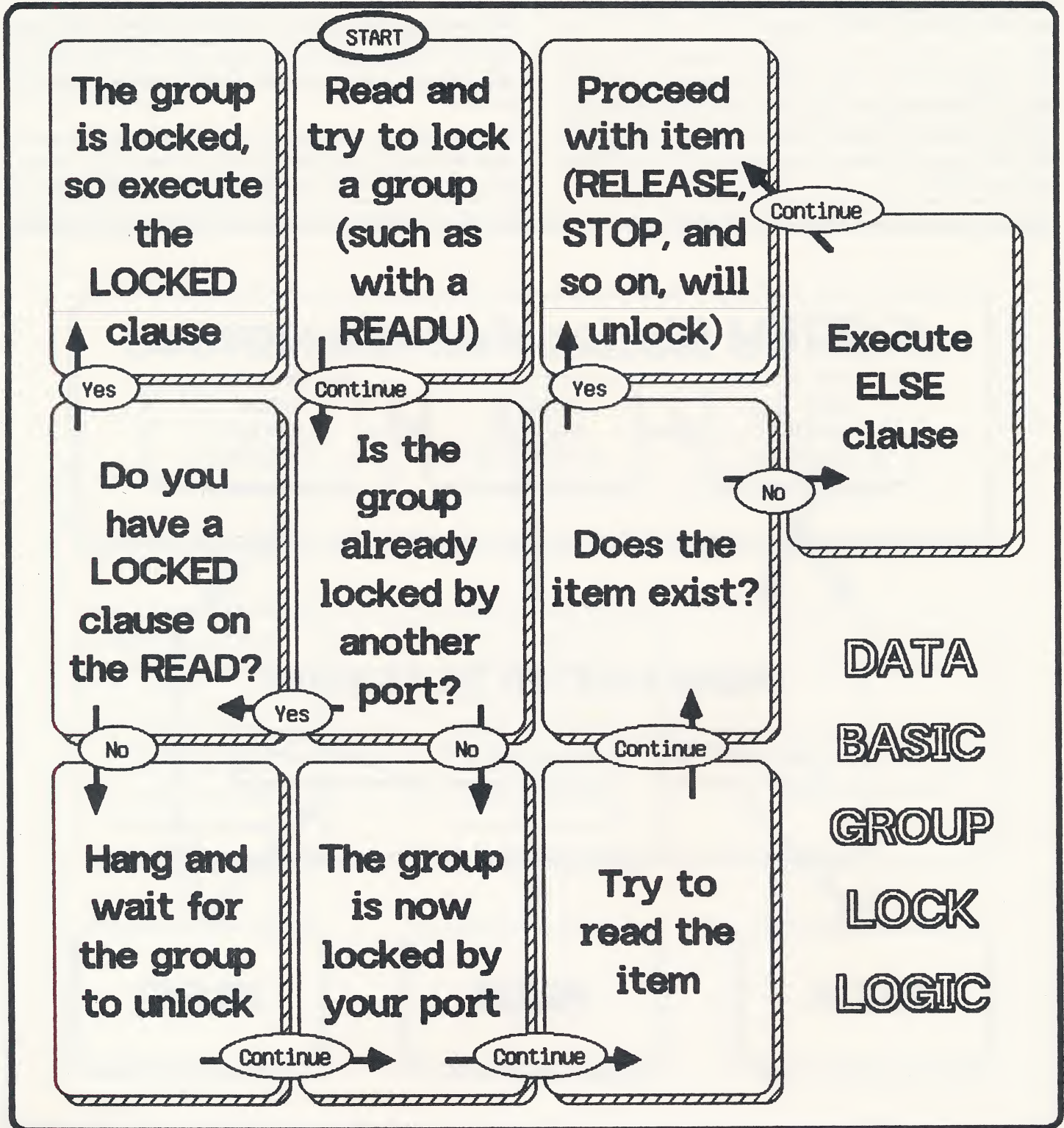
# Lock Logic Illustrated

A flow chart is presented to help explain the intricacies of statements like READU.

The diagram below is intended to help programmers understand the effect of DATA/BASIC™ statements that try to read an item and lock its group.

Note that programs that simultaneously lock multiple items *in the same file* may release a group prematurely if two locked items happen to fall in the same group. In such a case, the programmer might think that at least one item is locked at all times, but when the program does release one item, the program in effect releases both, since both items are in the same group.

2





## MICRODATA®

USED EQUIPMENT & SYSTEMS

- 50 MB REFLEX I, DISC DRIVES
- 50 MB REFLEX I with CONTROLLER
- 128 MB REFLEX II, DISC DRIVES (NEW)
- 128 MB REFLEX II, with CONTROLLER
- 128 K MOS MEMORY BOARDS
- 16K CORE MEMORY BOARDS
- 8-WAY BOARDS with PORTS, CABLES
- INTELLIGENT 8-WAY BOARDS
- PRISM I CRTS
- PRISM II CRTS
- ADDS VIEWPOINT CRTS (NEW)
- 300LPM PRINTRONIX PRINTERS (NEW)
- PRINTER CONTROLLERS with CABLE
- 600LPM PRINTRONIX PRINTERS (NEW)
- CORE SYSTEMS
- MOS SYSTEMS

CALL CHUCK HAAS FOR  
COMPETITIVE PRICES

BUY-TRADE-SELL

513-528-5400

PLEASE NOTE OUR NEW TELEPHONE NUMBER

ESYSPROG II



# the computer room

## Password Protection

Simply guessing passwords is the primary method used by unauthorized personnel when accessing supposedly "secure" computer systems.

Even when passwords are kept secret, they are often so simple that it is a trivial matter to find a password by brute force searching. For example, many computer systems come with a standard account supplied by the vendor. For Pick systems, this would be the SYSPROG account. A computer bandit typically knows the name of the standard account for the computer being burgled, so all that is left to do is guess the password. If a password is a single upper case character, a maximum of 26 tries (and just 13 on the average) are necessary to find the password. If the password is two upper case characters, a maximum of 676 tries is necessary, so having longer passwords obviously helps deter password searches. Unfortunately, bandits can easily write a program that automatically tries all password combinations! A computer system with a dial-up modem can be accessed by another bandit computer programmed to try and find passwords, and the computer system being accessed can't even tell whether a legitimate user has called and is trying to enter a password, or whether it is another machine.

A medium speed bandit computer can be programmed to try submitting all passwords of one upper case character in about 30 milliseconds. (That's 30 thousandths of a second, ignoring delays such as modem transmission speeds and prompt messages.) The same program can test all possible passwords of up to four upper case characters in just 10 minutes! Even five character passwords would take only about four hours to exhaust, which means a bandit with a computer and a modem could probably break into the vast majority of systems that can be dialed up, assuming the correct telephone number is available.

If you assign a password, make it at least six letters, so that brute force searching by the hypothetical bandit would take at least 100 hours, hopefully allowing enough time to detect that unauthorized access is being attempted. Better yet, don't restrict the password to only upper or lower case letters; use punctuation and control characters. A keyboard can typically generate 128 different key codes. If a four character password is allowed to contain those control characters, it will take well over 90 hours for the bandit computer to try all possible four character combinations, as opposed to just four minutes when only letters of one case are used.



## PHARMACY SOFTWARE

(To run on any Pick® system)

We knew what was needed when we developed it, because we're Pharmacists and users too. And that's why it is the most advanced and comprehensive Pharmacy software available.

**We have complete:**  
**Nursing Home Pharmacy Package**  
**Retail Pharmacy Package**

Dealer inquiry invited to set up on line and/or stand alone system.

Contact David N. Zimmerman, PD, FACA, NHA  
ALLIED COMPUTER SYSTEMS  
(A division of)  
ASSOCIATED PHARMACY CONSULTANTS, INC.  
13700 Woodward Ave.  
Highland Park, Michigan 48203  
(313) 869-1806



# Two Undocumented Conversions

Two conversion codes available on Microdata™ systems for testing the range of an integer and for testing the length of a string are generally unknown among users because for some reason the conversions have remained undocumented. A conversion such as OCONV(X, "R1,10") will return X if it is in the range from 1 to 10, otherwise the conversion returns a null. Similarly, a conversion like OCONV(X, "L1,10") will return X if its length is from 1 to 10 characters, otherwise null is returned. These conversions are similar, but not identical, to the L and R conversions available on other Pick systems. (The L0 form for returning the actual length does not appear to work on Microdata machines, both L arguments are always required, and the semi-colon option for repeating R ranges is not available.)

The three programs immediately below show simple tests of the conversions. The RANGE program will only output an input string that falls between 30 and 59, inclusive, which hints at how R conversions are particularly convenient for applications such as aging reports. It is interesting to note that an input of 40.1 is treated as though 40 were input. The LENGTH.CHECK1 program is a typical range check, conventionally programmed. LENGTH.CHECK2 shows how the same test can be coded in a slightly shorter fashion by using an L conversion. Benchmarks have shown that for short strings, the L conversion is about 15% faster than using the LEN function.

Following the test programs, an example of an invoice file dictionary is given, showing how R conversions might actually be used for aging definitions (assuming attribute five is a due date and attribute six is an amount due). The PRINT.AGING proc invokes the dictionary, and the listing that follows shows what the output for such an aging report would look like.

Thanks go to Ed Sheehan of Meyers & Associates for pointing out these undocumented features. Note that since capabilities such as these two conversions are usually undocumented because of bugs or because of no plans for future support, they should only be used with care.

## EXAMPLES OF L AND R CONVERSIONS IN ACTION.

```
RANGE
001 100 PRINT "INPUT STRING":
002 INPUT STRING
003 PRINT OCONV(STRING, "R30, 59")
004 GO TO 100
005 END
```

```
LENGTH.CHECK1
001 100 PRINT "INPUT STRING":
002 INPUT STRING
003 LENGTH = LEN(STRING)
004 IF (LENGTH < 3) OR (LENGTH > 5) THEN PRINT "WRONG LENGTH!"
005 GO TO 100
006 END
```

```
LENGTH.CHECK2
001 100 PRINT "INPUT STRING":
002 INPUT STRING
003 IF OCONV(STRING, "L3, 5") = "" THEN PRINT "WRONG LENGTH!"
004 GO TO 100
005 END
```

| INVOICES.. | S/NAME..    | CONV | CORR.....                               | . MAX |
|------------|-------------|------|-----------------------------------------|-------|
| 30. DAYS   | S 0 30 DAYS | MD2  | A; IF ((D-5)(R1, 29))#" THEN 6 ELSE ""  | R 10  |
| 60. DAYS   | S 0 60 DAYS | MD2  | A; IF ((D-5)(R30, 59))#" THEN 6 ELSE "" | R 10  |
| 90. DAYS   | S 0 90 DAYS | MD2  | A; IF ((D-5)(R60, 89))#" THEN 6 ELSE "" | R 10  |
| DATED      | S 5 DATED   | D2-  |                                         | R 10  |

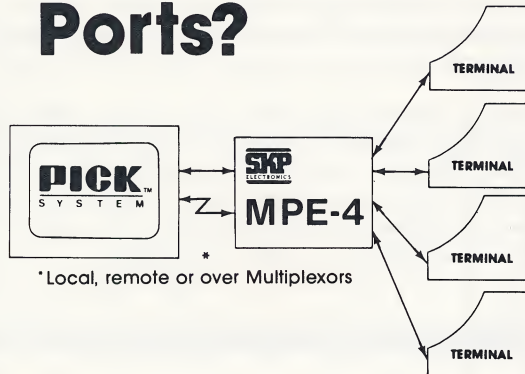
```
PRINT. AGING
001 PQ
002 H SORT INVOICES
003 H DATED
004 H 30. DAYS
005 H 60. DAYS
006 H 90. DAYS
007 H HEADING "PAGE 'P' - PRINT. AGING - 'TL'"
008 P
```

PAGE 1 - PRINT. AGING - 16:20:17 01 OCT 1983

| INVOICES.. | DATED..... | 30 DAYS... | 60 DAYS... | 90 DAYS... |
|------------|------------|------------|------------|------------|
| 2309       | 07-25-83   |            |            | 148.22     |
| 3428       | 08-10-83   |            | 432.83     |            |
| 4453       | 09-15-83   | 238.82     |            |            |



## Need More Ports?



The MPE/4 allows 4 terminals to share one CPU Port. All electronic switching with optional log off. For more information call or write:



1232-E South Village Way  
Santa Ana, CA 92705  
(714) 972-1727

Pick Systems is a trademark of Pick Systems, Inc.

### FOR SALE Microdata™ R6555

128K MOS Memory  
2 50MB Reflex® Disks  
2 8-ways  
1600BPI Tape  
300 LPM Printronix Printer  
2 Prism® CRTs

Price Negotiable  
Contact Jan Vath • 408-730-1825

### FOR SALE Microdata™ Reality®

64KB Memory  
20MB Disk  
800BPI Tape Drive  
8 Ports  
1 CRT Terminal

\$8,000 or Best Offer  
Call Sheila Sievers at 513-277-8933

## Letters

If you use a Pick system, Pragma wants to hear from you. Have you developed applications of interest to other users? Do you plan to acquire new hardware? What features would you like to see in Pragma? Are you active in any type of user group organization? Write Pragma today. All letters to the Editors are welcome, and as many as possible will be published in the Letters Department in every issue.

#### Does the Pick Operating System Need a Textbook?

Your publication has been very helpful to us. We have recently come to Microdata from Univac and IBM environments and are eager to see publications for Pick systems.

I am an instructor at Virginia Commonwealth University here in Richmond, and interested in putting together an introductory text for Microdata/Pick programming. During our start-up period on this machine, we were constantly frustrated by the lack of a good text on relational databases, proc, DATA/BASIC and ENGLISH. Do you feel that a text to fill this void could be successful?

Gerald A. Saunders  
Gerald Saunders Systems Design  
Richmond VA

*Yes, a book on the Pick operating system should do well, which is probably why we know of at least three individuals and/or companies that are currently trying to write one. Probably the toughest problem for any author working on such a book is deciding how to effectively deal with all of the differences between the various vendor implementations. The Pick system really needs two books: a "standard", detailed, generic reference manual and a tutorial. The operating system will probably never reach its full potential and market acceptance until both of those books are written.*

— Editors

P

### Problem? Inquiry? Idea? Complaint?

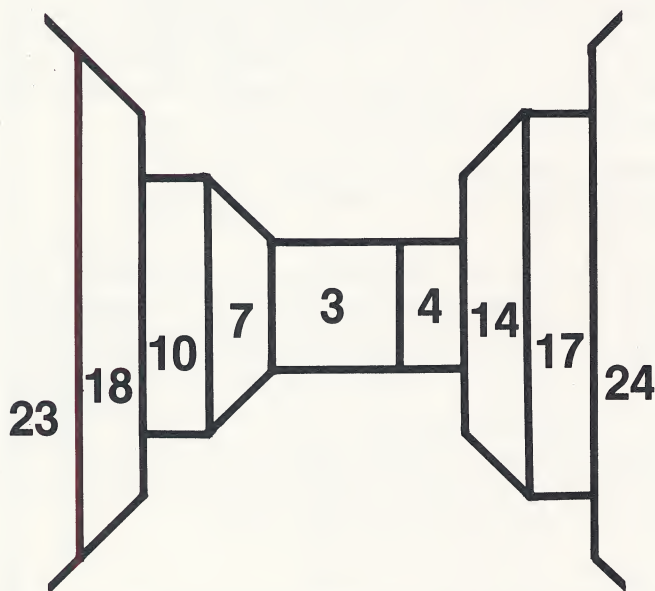
Write Pragma Today!



# games

## How AMAZING Works

You've been wandering through the labyrinth of BASIC statements for over three hours now, trying to understand the program's logic. Your CRT is displaying the view from your current position inside the data:



You are looking down a corridor that ends up turning right, but there is also a passage off to the left and an even nearer hall off to the right. How can this program possibly be changed to reveal a pile of gold at the end of a corridor?

For many people, computers are synonymous with games, now that the video game industry has become such a giant. Programmers frequently cut their teeth on small game programs, since such programs are often straight-forward and self-contained without a lot of complicated interfaces to files or other software. And, more than anything, games are simply entertaining and fun.

The Games Department will be making periodic appearances in *Pragma*. If you have a game program you would like to share, send it in for publication. If there's anything the *Pragma* staff has time to do, it's "performing rigorous software quality assurance" (in other words, trying out a new game on the computer).

The previous article in this department [*Pragma* #5, page 51] introduced the AMAZING program for maze exploration, and many readers have since asked for more details on exactly how AMAZING works. The two diagrams at the top of the opposite page should help clarify many of the mysteries concerning AMAZING's internal logic.

The upper left diagram shows the 82 different possible line segments that might be displayed by AMAZING. (Note that line 81 overlaps a portion of line 15, and line 82 similarly overlaps line 18.) These are the line segments which are stored as endpoint co-ordinates in the EDGES item read by AMAZING. In effect, these lines form the edges of 24 different square wall panels visible (in whole or in part) from the explorer's current position within the maze. Only 24 can be visible because AMAZING assumes there is always at least one wall blocking the view within four panels of the observer.

The upper right diagram is a "floor plan" indicating how each of the 24 walls is numbered. To understand how the two diagrams relate, verify that wall 3 is formed by edges 38, 44, 43 and 49, while wall 21 has edges 4, 6, 9 and 77. Each time the explorer moves, each element of the W array in the AMAZING program is set to true or false by subroutine 10, depending on whether the MAZE structure includes a corresponding visible wall at that position, relative to the explorer. The calculation is simplified by the OFFSETS item, which remembers all relative wall positions. For example, OFFSETS shows that wall 17 is two units to the right and three units up from the observer's position. (An OFFSETS unit is half the distance between walls.)

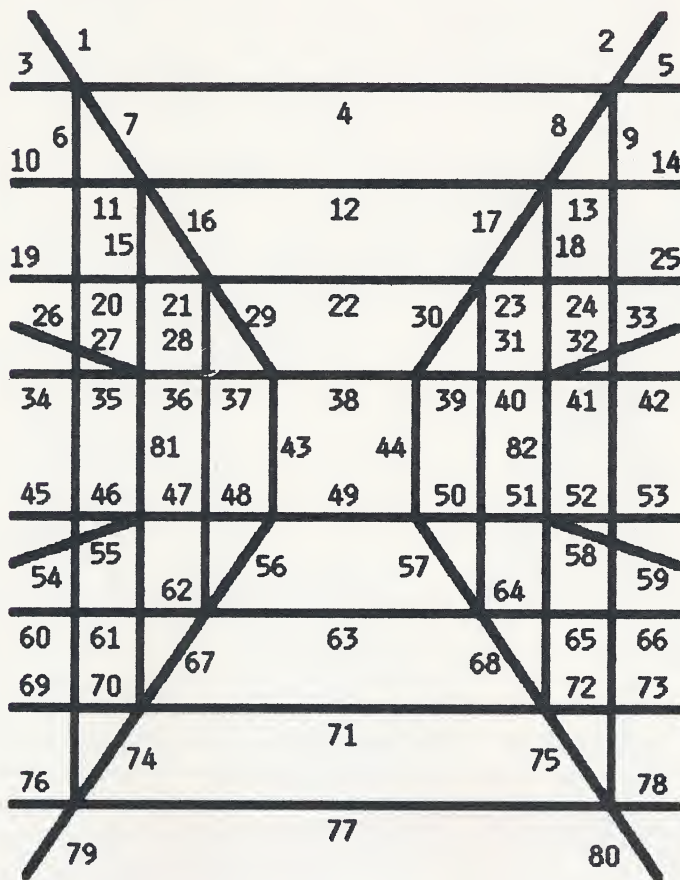
In order to generate the final display, AMAZING subroutine 30 simply tests each line segment and only displays it if it is part of an existing wall, and no other wall hides it from view. For example, line 94 in subroutine 30 only displays edges 12 and 71 if wall 16 is present and wall 21 is not.

The original article also briefly mentioned speeding up subroutine 20 to allow faster line plotting. The code shown below the two diagrams is just such an improved plot routine. Not only is it over 40% faster than the previously published version (while avoiding ICONV calls), it is also designed to handle lines of any slope. The old program assumed lines were being plotted from left to right and top to bottom.

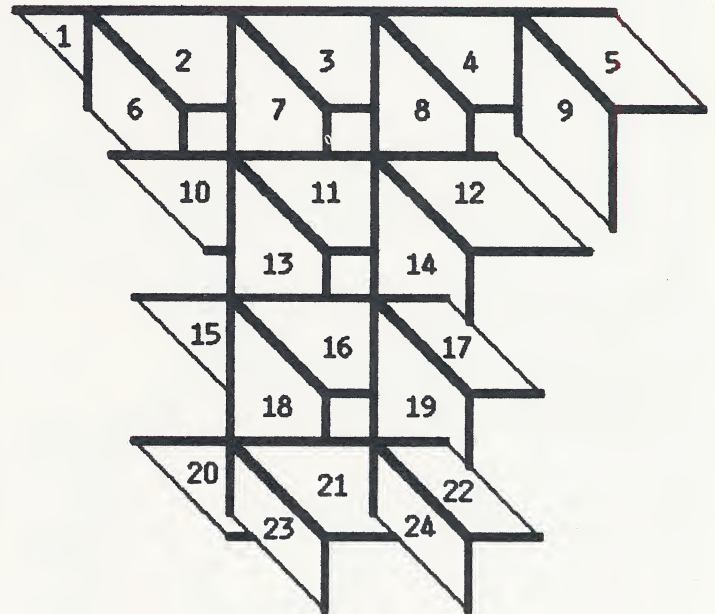
P



## ALL DISPLAYABLE LINES



## THE 24 POSSIBLE WALLS



```

001 20 * Plot line
002 XFROM = EDGES<L,1> ; YFROM = EDGES<L,2>
003 XTO = EDGES<L,3> ; YTO = EDGES<L,4>
004 XDELTA = XTO-XFROM ; YDELTA = YTO-YFROM
005 XINCB = 0 ; YINCB = 1 ; XINCA = 1 ; YINCA = 0
006 IF XDELTA < 0 THEN XINCA = -1 ; XDELTA = -XDELTA
007 IF YDELTA < 0 THEN YINCB = -1 ; YDELTA = -YDELTA
008 IF XDELTA < YDELTA THEN
009     SWAP = XDELTA ; XDELTA = YDELTA ; YDELTA = SWAP
010     XINCB = XINCA ; XINCA = 0 ; YINCA = YINCB ; YINCB = 0
011     END
012 DELTA = INT(XDELTA/2)
013 N = 1
014 LOOP
015     PRINT @(XFROM,YFROM): "+":
016 UNTIL N > XDELTA DO
017     XFROM = XFROM + XINCA ; YFROM = YFROM + YINCA
018     DELTA = DELTA + YDELTA
019     N = N+1
020     IF DELTA > XDELTA THEN
021         DELTA = DELTA-XDELTA
022         XFROM = XFROM + XINCB
023         YFROM = YFROM + YINCB
024     END
025 REPEAT
026 RETURN
    
```

## AN ALTERNATIVE LINE PLOTTER



# PRAGMA

207 GRANADA DRIVE  
APTOS, CA 95003

BULK RATE  
U.S. POSTAGE  
PAID  
APTOS, CA 95003  
Permit No. 67

... Address Correction Requested

## SEMAPHORE CORPORATION

### Provides Software and Support for Microdata Systems

- COMPLETE SPECIFICATION
- ELEGANT DESIGN
- SYSTEMATIC IMPLEMENTATION
- THOROUGH TESTING
- PATIENT USER TRAINING
- ONLINE DOCUMENTATION
- PROVEN POLICIES AND PROCEDURES

## SEMAPHORE

SEMAPHORE CORPORATION  
207 GRANADA DRIVE  
APTOS, CA 95003  
408-688-9200

- SPECIALIZING IN MANUFACTURING SYSTEMS:** ENGINEERING • PURCHASING  
• RECEIVING • INSPECTION • ORDER ENTRY • SHIPPING • CUSTOMER SERVICE  
• PRODUCTION CONTROL • SHOP FLOOR • PAYROLL • PERSONNEL  
• RECEIVABLES • PAYABLES • COST ACCOUNTING • GENERAL LEDGER  
• FIXED ASSETS • MODELING